Faculty of Information Technology

Master of Computing

Reducing Test Power for Embedded Memories

Prepared By

Ahmed Awad

Supervised By

Dr.Abdellatif Abu-Issa

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF COMPUTING

BIRZEIT

June, 2011

Acknowledgement

I would like to acknowledge and extend my heartfelt gratitude to all those who helped and supported me to complete my thesis. I'm deeply indebted to my supervisor Dr.Abdellatif Abu-Issa for his help, support and continuous encourage for me. Also, special thanks to the committee of my master thesis defense, Dr.Emad Hamadeh and Dr.Khaled Faraj for their comments to have a high quality thesis. I want also to thank Dr.Said Hamdoui from Delft University in Netherlands for his guidance during working on conference papers. Most special thanks to my father and mother for their support and constant prayers and my family and all my friends who love and encourage me. Thank you all.

Table of Contents

List of Figures ..... **IV**
List of Tables ..... **V**
List of Abbreviations ..... **VI**
Abstract ..... **VII**
المستخلص ..... **VIII**
Chapter 1: Introduction **1**
1.1 SRAM Structure and Functionality **2**
1.2 SRAM Fault Models **6**
1.3 Memory Testing Concepts **9**
1.4 Memory Testing Algorithms **12**
1.5 Power Dissipation in SRAM Testing **15**
1.6 Motivation for SRAM Test Power Reduction **17**
1.7 Thesis Organization **18**
Chapter 2: Related Work **19**
2.1 Single Bit Change (SBC) in Address Decoder **19**
2.2 Minimizing Test Power Through Reduction of Pre-Charge Activity **21**
2.3 March Tests Sequence Reordering Using Genetic Algorithm **22**
2.4 Generating Low Power March Tests Using Particle Swarm Optimization **23**
2.5 Skew Scheme **25**
2.6 Power Constrained Embedded Memory BISTArchitecture **26**
2.7 Contribution of The Work Presented In This Thesis **28**
Chapter 3: Low Power Zero-One Testing **29**
3.1 Overview **29**
3.2 MBIST Architecture and Address Generators **30**
3.2.1 Bit-Swapping Lfsr (BS-LFSR) **31**
3.2.2 Dual Speed Lfsr (DS-LFSR) **32**
3.2.3 Bipartite LFSR **33**
3.3 Detecting Stuck-At Fault Patterns **34**
3.4 Simulations and Experimental Results **35**
3.4.1 Code Description **36**
3.4.2 MBIST Simulation **36**
3.4.3 Address Generators Simulations and Results **37**
3.4.4 Testing Patterns Simulations and Results **39**
3.5 Summary **40**
Chapter 4: Low Power March Tests **41**
4.1 Overview **41**
4.2 Modified March C- Algorithm **42**
4.3 Implementation **44**
4.4 Fault Coverage **45**
4.5 Experimental Results **47**

4.5.1 Power Estimation.....47
4.5.2 Simulation Results 47
4.6 Summary 49
Chapter 5: Low Power Schemes For Parallel Testing of Embedded Memories .50
5.1 Overview 50
5.2 One-Stage Scheme 51
5.3 Multi-Stage Scheme 53
5.4 MBIST Implementation 56
5.5 Experimental Setup and Simulation Results 57
5.6 Summary 59
Chapter 6: Conclusions and Future Work 60
6.1 Conclusions.....60
6.2 Future Work 61
References.....62
Appendix A 67
Appendix B 69
Appendix C 70
Appendix D 71

List of Figures

Figure 1.1: Memory Pyramid 3
Figure 1.2: Increased number of SRAMs in SoC 3
Figure 1.3: 6T SRAM cell 4
Figure 1.4: SRAM Connections 5
Figure 1.5: SRAM cell during Read 1 Operation 5
Figure 1.6: SRAM cell during Write 0 Operation 7
Figure 1.7: Stuck-at 0 cell State Diagram 7
Figure 1.8: Address Decoder Fault 7
Figure 1.9: Transition Fault State Diagram 7
Figure 1.10: Coupling Fault State Diagram 9
Figure 1.11: Memory Testing Pattern 10
Figure 1.12: MBIST Architecture 11
Figure 1.13: 3-bit LFSR 12
Figure 1.14: Symbols Used in Memory Testing Algorithms 13
Figure 1.15: Zero-One Algorithm 14
Figure 1.16: March C- Algorithm 14
Figure 1.17: First two Elements of March C- 15
Figure 1.18: CMOS Logic 15
Figure 2.1: Modified Pre-charging Circuitry 22
Figure 2.2: Switching Activity in March C- Algorithm 22
Figure 2.3: PSO Particle Structure 24
Figure 2.4: Applying element M1 of March C- on two SRAMs 25
Figure 2.5: Skew Scheme 26
Figure 2.6: Power-Constrained MBIST Architecture 27
Figure 2.7: Wrapper Address Generator 27
Figure 3.1: Used MBIST Architecture 31
Figure 3.2: BS-LFSR 31
Figure 3.3: DS-LFSR 32
Figure 3.4: Intermediate Pattern Generation in Bipartite LFSR 33
Figure 3.5: Bipartite LFSR Architecture 33
Figure 3.6: Fault Free Memory Simulation 37
Figure 4.1: March C- Algorithm 42
Figure 4.2: Modified March C- Algorithm 43
Figure 4.3: Tri-State Buffers for 4-bit word 45
Figure 4.4: Transitions of two vertically neighbored cells in March C- 46
Figure 4.5: States of two vertically Neighbored Cells in Modified March C- 46
Figure 4.6: States of two horizontally neighbored cells in Modified March C- 46
Figure 4.7: Expanded March C- Algorithm 47
Figure 4.8: States of two vertically neighbored cells in Expanded Test 47
Figure 4.9: Transitions of two horizontally neighbored even cells in Expanded Test 48
Figure 5.1: One-Stage Scheme 51
Figure 5.2: Two-Stage Clustering 53
Figure 5.3: Multi-Stage Scheme 55
Figure 5.4: Architecture of Low Routing MBIST 56
Figure B.1: Crossover Operation 69
Figure D.1: Maximal Length LFSR 71

List of Tables

Table 2.1: SBC in Address Decoder 20
Table 2.2: New Generated March Tests using Genetic Algorithm 23
Table 2.3: New March Tests Based on PSO Scheme 25
Table 3.1: Normal and BS- LFSR Vectors 32
Table 3.2: Switching Activity for Address Generators with Seed “1111….1” 38
Table 3.3: Switching Activity for Address Generators with Seed “0101…..01” 38
Table 3.4: Address and Data Bus Switching Activities for Different Patterns.....39
Table 4.1: Power results of normal, modified and expanded March C- Tests.....49
Table 5.1: Used Memory Configurations 57
Table 5.2: Peak Power and Testing Time for Different Schemes.....58
Table 5.3: Combining One-Stage Scheme with Modified March C- Algorithm.....59
Table A.1: Fault Primitives 67
Table A.2:Some March Tests with their Fault Coverages 68
Table D.1: Maximal Length LFSRs.....71

List of Abbreviations

- 1T : One Transistor Technology
- 6T : Six Transistor Technology
- ATE : Automatic Test Equipment
- BIST : Built-in Self Test
- BL : Bit Line
- BLB : Bit Line Complement
- BS-LFSR : Bit-Swapping LFSR
- CMOS : Complementary Metal Oxide Semiconductor
- CUT : Circuit Under Testing
- DRAM : Dynamic Random Access Memory
- DRDF : Deceptive Read Destructive Fault
- DS-LFSR : Dual-Speed LFSR
- IRF : Incorrect Read Fault
- LFSR : Linear Feedback Shift Register
- MBIST : Memory Built-in Self Test
- MUT : Memory Under Testing
- NOP : Number Of Operations
- PSO : Particle Swarm Optimization
- RDF : Read Destructive Fault
- RES : Read Equivalent Stress
- SA : Switching Activity
- SAIF : Switching Activity Interoperation Function
- SBC : Single Bit Change
- SoC : System on Chip
- SRAM : Static Random Access Memory
- TPG : Test Pattern Generator
- VLSI : Very Large Scale Integration
- WDF : Write Disturb Fault
- WL : Word Line

Abstract

With the increased number of embedded memories in mobile devices, minimizing the test power becomes a serious concern, especially when parallel testing is applied. Battery will be lost and the entire System on Chip (SoC) is subjected to be damaged if the consumed power exceeds the power constraint of the chip.

This dissertation proposes a number of techniques to address these challenges during memory testing. The first technique is based on using low power Linear Feedback Shift Register (LFSR) as an address generator when applying Zero-One algorithm during Memory Built-in Self Test (MBIST), and then, re-order the test so that total switching activity in address decoder and write driver is minimized. The obtained results show that up to 60% reduction in switching activity can be achieved during testing large size memories with negligible overhead in hardware area.

Another technique that aims to reduce average and peak power during March tests is proposed. In this technique, the word of the Memory Under Testing (MUT) is divided into two clusters so that write operation is applied just to one cluster. Obtained results show that around 42% reduction in peak power and around 35% reduction in average power can be achieved using the proposed technique with the same fault coverage and testing time of original tests.

Finally, a new scheme is proposed to manage parallel testing of large number of embedded memories in SoC. This scheme is based on grouping different memories into clusters based on their word lengths and scheduling read and write operations in such a way that the consumed power is optimal. Simulation results of case-of-study show that up to 60% reduction in peak power can be achieved in case of parallel testing at a cost of only one additional clock cycle in testing time

المستخلص

مع التزايد المستمر في عدد الذاكرات المستعملة في الأجهزة المتنقلة، تقليل القدرة المستهلكة عند فحص هذه الذاكرات أصبح تحديا هاما، خصوصا عند تطبيق الفحص المتوازي على هذه الذاكرات. طاقة البطارية سوف تستنزف والنظام على القطعة الإلكتروني ة بكامله معرض للتلف في حال تجاوزت القدرة المستهلكة الحد المسموح.

تعرض هذه الرسالة عددا من الطرق لمواجهة مشكلة القدرة المستهلكة أثناء فحض الذاكرة. تعتمد الطريقة الأولى على استعمال مولد للعناوين ذو طاقة قليلة أثناء استعمال طريقة صفر-واحد في فحص الذاكرة، ثم تقوم هذه الطريقة على إعادة ترتيب الفحص بطريقة تقلل التغيرات في القيم الرقمية إلى الحد الأدنى. وقد أثبتت النتائج يمكن حفظ حوالي 60% من القدرة المستعملة باستخدام هذه الطريقة مع زيادة لا تذكر في المساحة في القطعة الالكترونية.

تعتمد الطريقة الثانية على تقليل القدرة المتوسطة والقدرة القصوى إثناء استعمال الفحص الشامل للذاكرة لكشف أنواع كثيرة من الأعطاب. يتم ذلك بواسطة تقسيم الخلايا في كل عنوان في الذاكرة كل مجموعتين، ومن ثم القيام بعملية الكتابة فقط على مجوعة واحدة من تلك المجموعات. وقد أثبتت النتائج انه يمكن حفظ حوالي 42% من القدرة القصوى وحوالي 32% من القدرة المتوسطة باستخدام هذه الطريقة مع كشف نفس الأنواع من الأخطاء وفي نفس زمن الفحص

وأخيرا، تقوم الطريقة الثالثة على تقليل القدرة القصوى المستهلكة عند فحص عدد كبير من الذاكرات بالتوازي. وتقوم هذه الطريقة على توزيع الذاكرات في مجموعات بناء على أحجامهم ومن ثم جدولة عمليات القراءة والكتابة بطريقة تقلل القدرة القصوى إلى الحد الأدنى. النتائج أثبتت أنه يمكن حفظ أكثر من 60% من القدرة القصوى مع زيادة مهملة في وقت الفحص.

CHAPTER 1

Introduction

With the advances in Very Large Scale Integration (VLSI) technology, more and more contents are integrated together in System on Chip (SoC).Semiconductor embedded memories can be considered as the densest circuitry and it is expected that in 2014, embedded memories will occupy around 94% of silicon area in the SoC [1]. Due to their high density and intensive access, embedded memories are more likely to be affected by manufacturing faults rather than other components in the chip. Hence, memories have to be tested effectively [2].

Due to its high speed and reliability, Static Random Access Memory (SRAM) is more commonly used in different applications such as digital cameras and mobile phones, for this reason, many techniques were developed for testing embedded SRAMs [3]. The increased number of embedded memories in SoC makes testing process more complex in terms of time and power [4]. Testing power plays an important role in evaluating the effectiveness of the test. If the power consumed exceeds the accepted power constraint, then the chip is subjected to structural degradation and may be damaged [5].

This thesis addresses the problem of power consumption during testing SRAM for the existence of manufacturing faults. It studies the existing algorithms for memory testing and provides an enhancement for some of those techniques so that testing power is reduced with the same fault coverage and within an accepted testing time. The main objectives of this dissertation are:

1. To reduce testing power of Zero-One algorithm that is used for testing embedded memories of personal applications such as digital cameras.

2. To reduce testing power of March tests that are used for intensive testing of embedded memories used in critical fields such as military applications.

3. To reduce peak power when parallel testing is applied for a large number of embedded memories in SoC.

This chapter introduces the main concepts about SRAM testing and manufacturing faults. Then it addresses memory testing patterns and the main sources for power dissipation during testing. Finally, the motivation of this dissertation and its general organization are summarized.

1.1 SRAM Structure and Functionality

SRAM is a volatile memory that is used to store binary values in computer systems. It stores each bit effectively using a latching circuit which is made of transistors. It loses its data when its power supply is turned off [6].

SRAM is faster than Dynamic Random Access Memory (DRAM) since no refreshment is required for its functionality. Actually, SRAM was found to reduce the gap in speed between Microprocessor Unit (MU) and the main memory in which DRAM is used. Thus, SRAM is used in the cache memory of computer. More than one level of cache may be required [7]. Figure 1.1 shows the memory pyramid which proves that memory was the driving factor behind the rapid development in Complementary Metal Oxide Semiconductor (CMOS) technology [8].

SRAM consumes less power than DRAM since DRAM refresh current is much higher than SRAM standby current. Also, it is more reliable than DRAM. Due to those advantages, SRAM is commonly used as an embedded memory in small and portable devices such as mobile phones and digital cameras. Also it is widely used in the buffers of routers and switches in the network. LCD screens and printers use SRAM to hold image that has to be displayed or printed [9].

The main disadvantage of SRAM is its high cost and the large area it occupies in SoC .Thus, large number of small SRAM memories is used in SoC instead of using large size ones .According to Moores' Law, the number of components on chip doubles every 18 months. Thus, the number of embedded SRAMs in SoC increases rapidly with time. Figure 1.2 shows how the number of SRAMs in SoC increases if compared with other logic on chip [11].

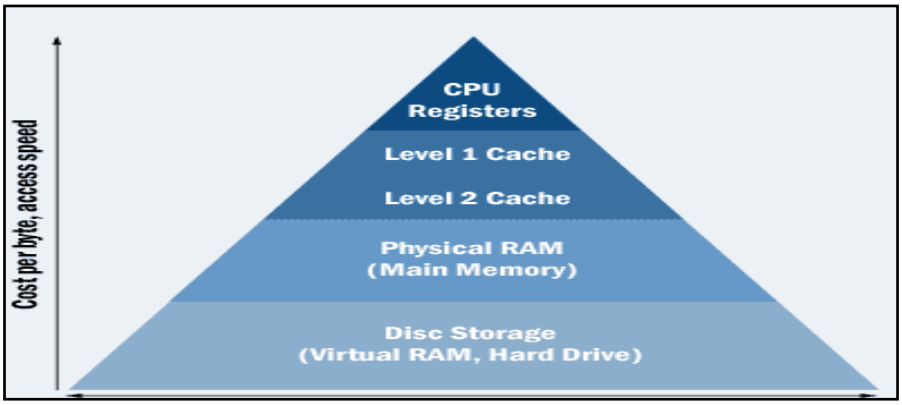

Figure 1.1: Memory Pyramid [10]

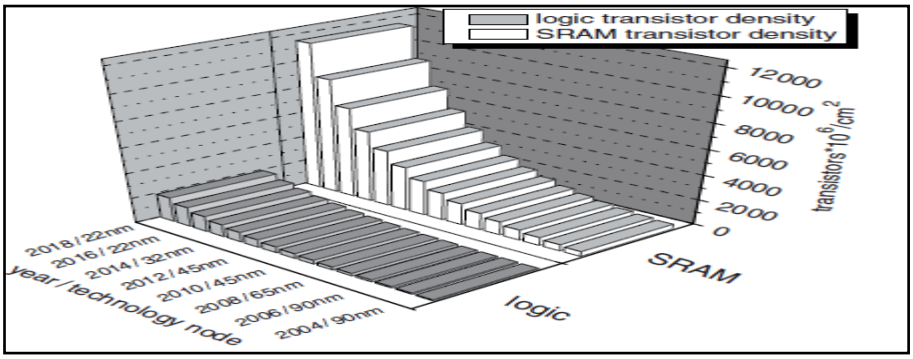

Figure 1.2: Increased number of SRAMs in SoC [11]

SRAM consists of a number of cells; each cell stores a binary value. There are several technologies for SRAM cells. To reduce the area occupied by embedded memories, one transistor (1T) technology, in which the cell consists of one transistor surrounded by intelligent control circuitry, was found and recently some companies have started to use this technology in the design of SoC [12]. However, six transistors (6T) SRAM cell is still the dominant technology used due to its reliability and stability and it is still considered in researches of test power, hence, it will be considered in this dissertation. As shown in figure 1.3, 6T cell consists of 4 transistors that forms two crossed coupled inverters to store the binary value. Two access transistors (Q5 and Q6) connect or disconnect the cell to the bit lines (BL, BLB). [13].

Usually, SRAM consists of a number of locations, according to the number of cells in each location; SRAM can be classified in two main types [14]:

1. **Bit Oriented SRAM** which contains only one cell in each memory location.
2. **Word Oriented SRAM**: In which each location contains a number of cells based on the word length. For example, if the word is 8 bit, then each memory

location contains 8 cells. Most of the memories used in SoC are word oriented ones with word lengths vary between 32-512 bits and 640bits in some cases such as video applications [15].

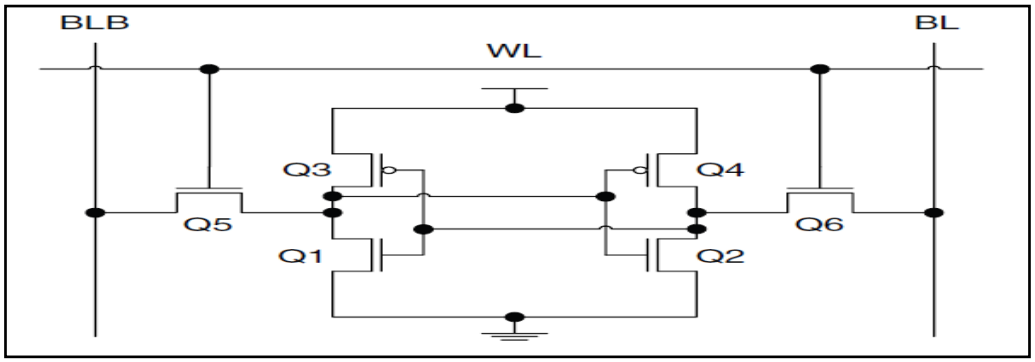

Figure 1.3: 6T SRAM cell [13]

SRAM is accessed by applying read and write operations on its cells. In case of read operation, the current value of the cell is retrieved whereas a new value is applied to this cell during write operation. Usually, address decoder is used to select the memory location that will be accessed. Then, the word line for that location is asserted so that the required cell is connected to the bit lines through access transistors. A control signal is used to determine whether the operation applied is read or write. In case of write operation, the value in the data bus is written to the cell. Some peripheral devices are required such as write driver which pulls down one of the bit lines so that the value in the data bus to be written is applied on the bit lines, and the sense amplifier that amplifies the small analog differential voltage developed on the bit lines by read operation to full swing digital output [13]. Figure 1.4 shows SRAM with its peripheral devices.

To apply a read operation, both bit lines are first pre-charged to Vdd. Then, the word line for the word that has to be accessed is asserted. By this way, the gate of the two access transistors (Q5 and Q6) is connected to logic 1. Consequently, both access transistors are ON and connect the cell to the two bit lines. If the cell contains logic 1, BL remains in its pre-charging level while BLB is discharged through transistors Q5 and Q1 that form a voltage divider whose output is no longer 0 and it is connected to the input of inverter Q2-Q4. In general $0+\Delta V$ should not exceed the switching threshold of the inverter Q2-Q4. Figure 1.5 illustrates a simplified model of 6T cell during read 1 operation. If the cell contains 0, BL will be discharged through

transistors Q6 and Q2 while BLB will remain in its pre-charging level, then the same operations will be applied [11].

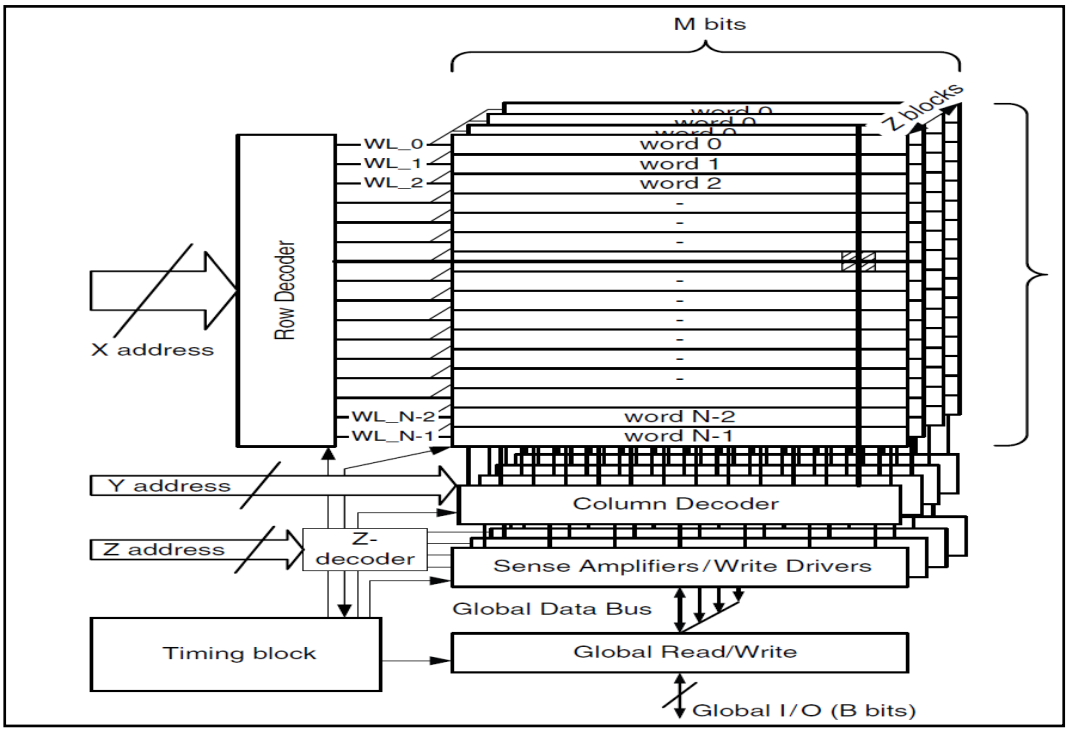

Figure 1.4: SRAM Connections [11]

In case of write operation, both bit lines are pre-charged to Vdd, then, one of those bit lines is pulled down through the strong write driver so that the value that has to be written is applied on both bit lines. In general, during write operation, the bit lines are driven by strong write driver to ensure overriding the current value that the cell holds. Thus, the write current is much higher than read current. Figure 1.6 shows a simplified model for SRAM cell when write 0 operation is applied [11].

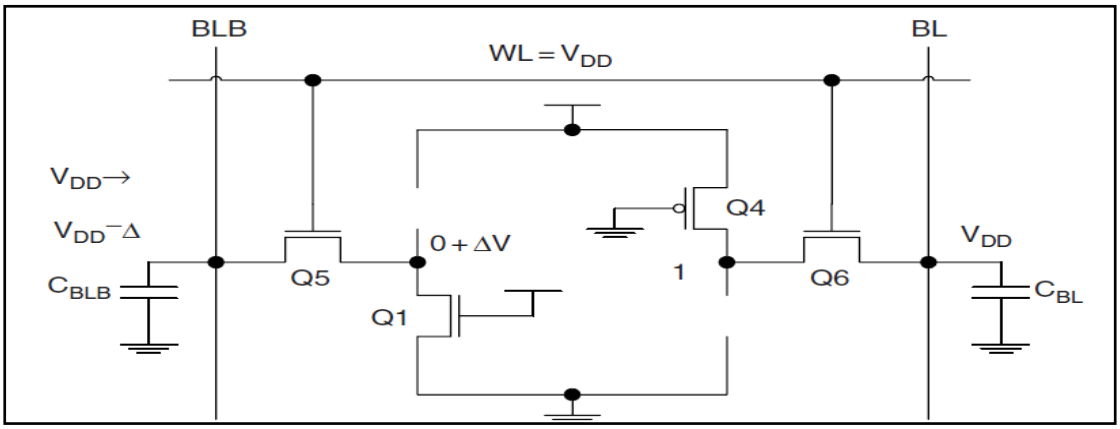

Figure 1.5: SRAM Cell during Read 1 Operation [11]

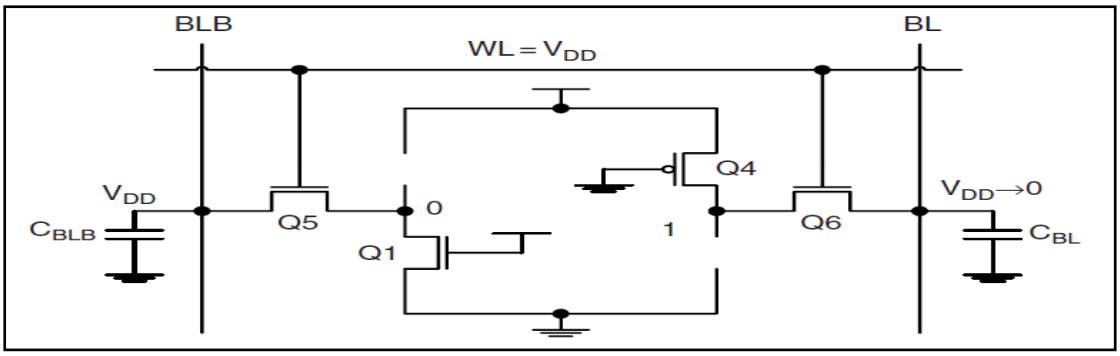

Figure 1.6: SRAM Cell during Write 0 Operation [11]

1.2 SRAM Fault Models

A fault model is a description about how the components of a faulty circuit will behave [16]. Due to the high density and intensive access of embedded SRAM instances in SoC, they are more likely to be affected by manufacturing faults that may affect the functionality of the system. The number of memory faults is more than that for any other component in SoC, for this reason several fault models were dedicated in order to detect these faults. [17]

SRAM faults can be classified in two main categories:

A. Single –cell Fault: A fault that involves only one cell in the memory, this includes the following faults:

1. Stuck-At fault: It is the most common type of faults in SRAM cells. In this fault, the cell value is stuck-at 0 or stuck-at 1 so that it cannot be overwritten by the complementary value. Typically, memories of personal devices such as mobile phones are tested mainly for this type of faults. Stuck-at faults can be caused by several reasons, such as short circuits in the connections and transistor defects. Figure 1.7 shows the state diagram of a stuck-at 0 cell. W0 and W1 represents write 0 and write 1 operation respectively while R means a read operation. S represents the current value of the cell [18].
2. Stuck open fault: In this type of faults, the cell cannot be accessed due to open word line or open bit line [19].
3. Address decoder fault: This type of faults can be classified under the stuck-at faults since one of the nodes in the address decoder may stuck

at 0 or 1 leading to accessing wrong address, no address, or multiple addresses. Figure 1.8 illustrates this type of faults [20].

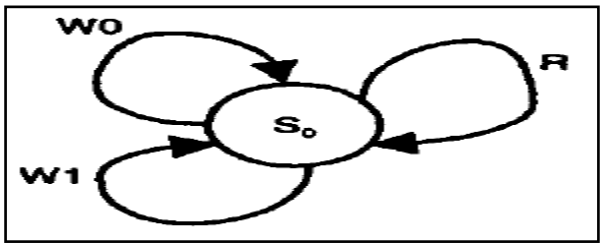

Figure 1.7: Stuck-at 0 cell State Diagram [18]

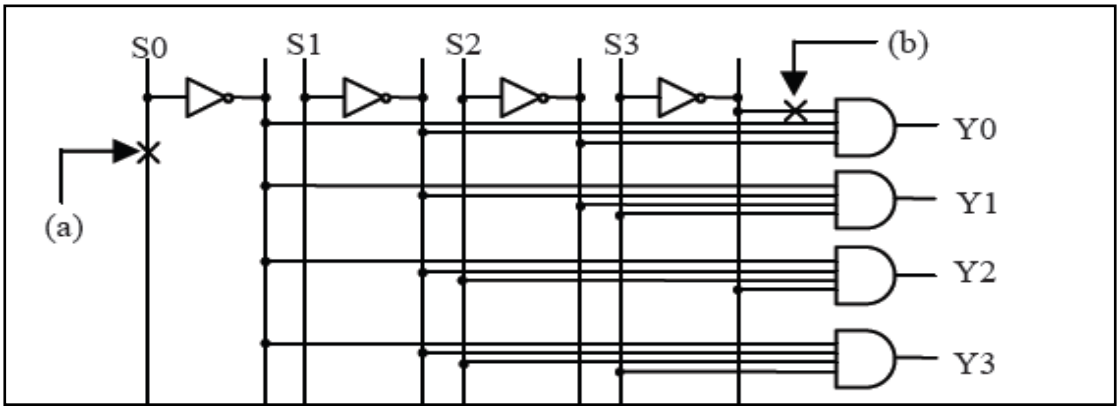

Figure 1.8: Address Decoder Fault [20]

4 Transition fault: In this type of faults, if the cell moves from one state to another, it cannot move back to the pervious state. For example, if 1 was written to a cell that contains 0, then 0 was written again, the cell state will remain 1. This type of faults can be caused by the absence of access transistors. Figure 1.9 shows the state diagram of this type of faults [18].

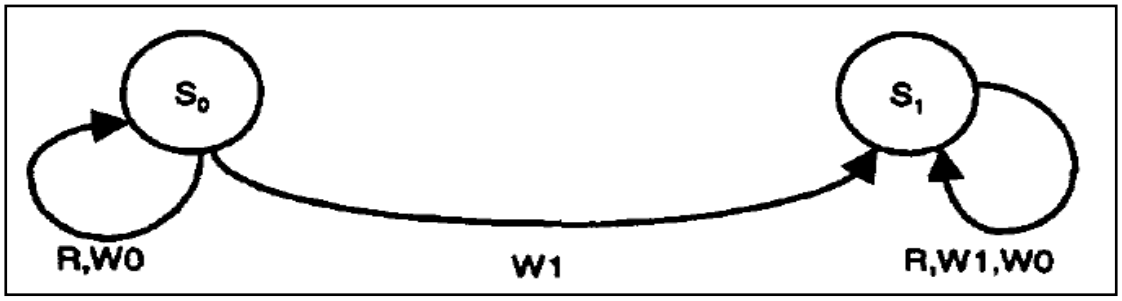

Figure 1.9: Transition Fault State Diagram [18]

5. Write Disturb Fault (WDF): In this fault, a non transition write operation will cause the cell to go into erroneous state. For example, if a cell contains 0 and 0 was written to it, then its state will be 1 [21].

6. Read Destructive Fault (RDF): The read operation in this fault will cause the cell to go into erroneous state. The result of read operation will be wrong [21]
7. Deceptive Read Destructive Fault (DRDF): This type of fault is the same as RDF but it is more difficult to detect since read operation result will be correct but the state of the cell will be wrong after this read operation [21].
8. Incorrect Read Fault (IRF): In this fault, the state of the cell will not be changed during the read operation, but the read result will be wrong [21].

B. Two-Cell Fault: A fault that involves two cells. This includes the following types of faults [22]:

1. Inversion coupling fault: In this type of faults, if the value of a cell is changed, then its neighbor will go into erroneous state. Usually the first cell is called **aggressor cell** whereas the affected cell is called **victim cell.** This fault could be symmetric; in which the victim cell state will go from high to low or from low to high following the aggressor cell transitions or it could be asymmetric so that the victim cell moves into one transition only. Also this fault could be one-way, in which the fault is sensitized by high to low or low to high transition or two-way which is sensitized in both transitions in the aggressor cell. Figure 1.10 shows the state diagram for a pair of cells with inversion coupling fault.
2. State coupling fault: Within this fault, the state of the aggressor cell will cause the victim cell to be affected by any of the previously mentioned single-cell faults. For example, if the aggressor cell state is 0, then the victim cell may be affected by transition fault.

C. Other faults: There are other types of faults such as delay related faults, stability faults and data retention faults. These types of faults are usually ignored in normal applications but have to be detected in applications that contain critical data in which memory has to be intensively accessed [23].

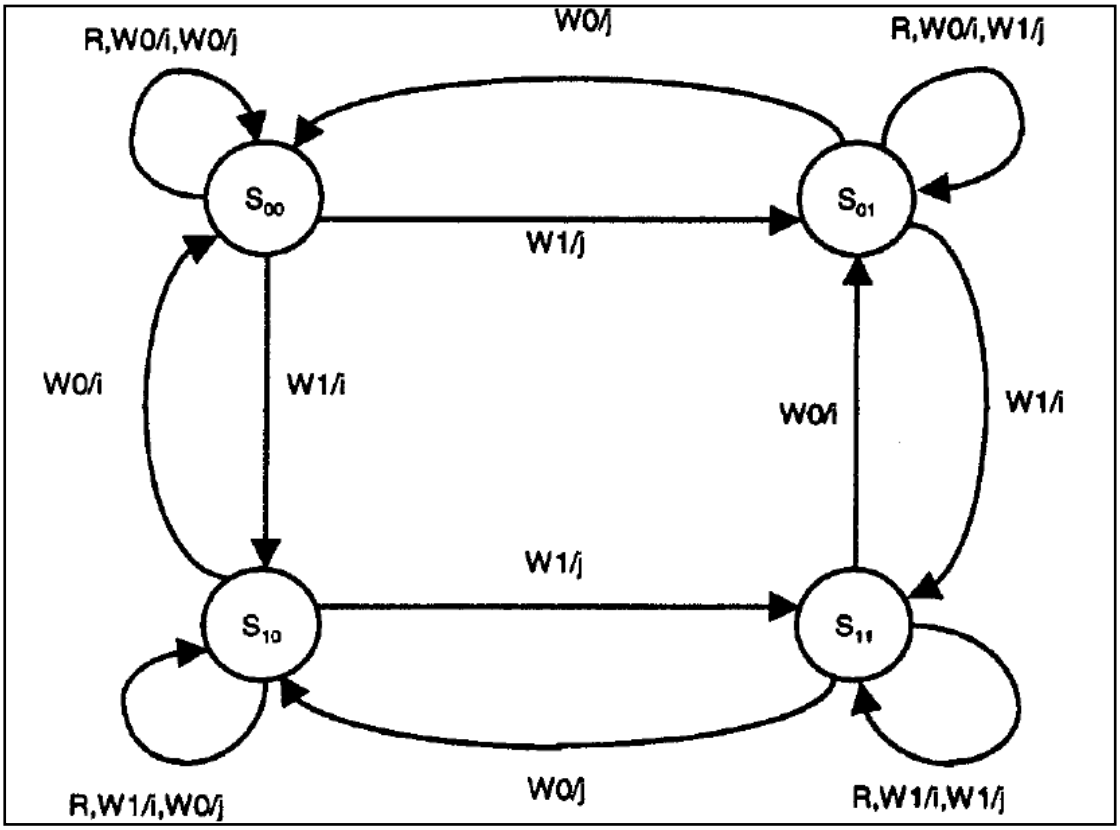

Figure 1.10: Coupling Fault State Diagram [18]

1.3 Memory Testing Concepts

SRAM has to be tested for the existence of manufacturing faults since these faults will affect the functionality of the memory. Testing phase means applying a set of patterns generated from a Test Pattern Generator (TPG) to the Memory Under Testing (MUT) and then comparing the obtained result with the expected result in case of fault free memory. **Testing pattern** consists of three parts as shown in figure 1.11 [24]:

1. The address that has to be accessed.
2. The data that will be written in case of write operation.
3. The control signal that determines whether a read or write operation has to be applied.

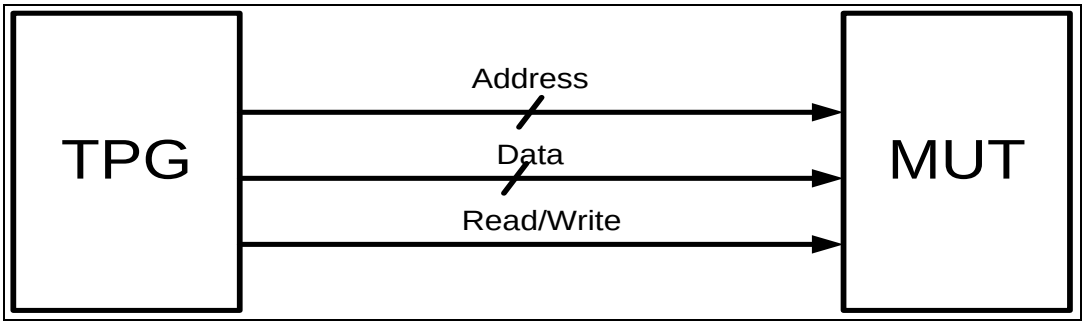

Figure 1.11: Memory Testing Pattern

Usually testing is performed either during manufacturing or during application. In general, testing is classified in two types:

1. External Testing: In this type of testing, an external instrumentation is used for generating testing patterns that are applied to the MUT. The equipment used is called Automatic Test Equipment (ATE). The main problem with this type of testing is its cost which is proportional to the number of pins of the MUT. Another disadvantage is the large testing time required since the SoC consists of hundreds of embedded memories that will be tested sequentially using a single ATE [25].

2. Built-in Self Test (BIST): this type is the most commonly used in testing memories. In Memory BIST (MBIST), a TPG is built in the chip that contains the MUT so that the memory is tested without any communication with the external world. BIST was found to overcome the problems of the external testing such as the high cost and the testing time. By using BIST, multiple embedded memories in SoC can be tested in parallel so that testing time is reduced. Figure 1.12 shows the main components of MBIST [26] which are:

1. BIST engine that generates the pattern (address, data, and control) that will be applied to MUT.

2. MUT that has to be tested for faults.

3. Comparator that is used to compare the results read from MUT with the expected result so that a pass/fail indication is generated.

4. Multiplexers whose selection line is the testing mode signal to determine whether the system is in the testing mode or in the normal mode. If the system is in the testing mode, then the pattern generated by the BIST engine will be applied to the memory, otherwise, the data coming from microprocessor will be the input of memory.

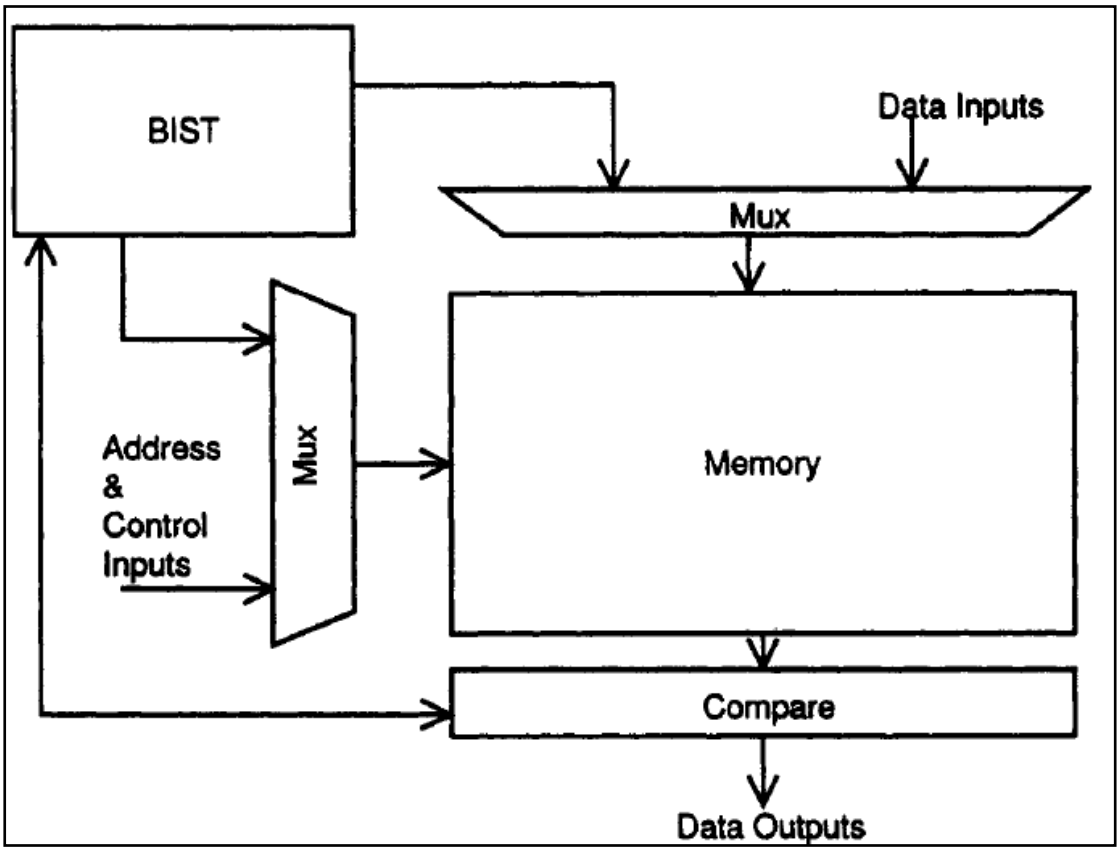

Figure 1.12: MBIST Architecture [18]

BIST engine consists of two parts which are: the **address generator** and the **controller**. The address generator generates the address that has to be accessed. Usually a counter is used as an address generator, but this will result in large overhead in the hardware area, thus, **Linear Feedback Shift Register (LFSR)** is sometimes used instead of the counter due to its low overhead in the hardware area [27]. LFSR is a register that consists of a number of flip flops and an XOR gate that is located based on the characteristic polynomial. It can be considered as a source of binary pseudorandom test sequences that can be used in testing combinational circuits and also as an address generator in memory testing. LFSR can be maximal length if it generates all the possible testing vectors (except the 0's vector since it blocks the LFSR). Figure 1.13 shows 3-bit LFSR with characteristic polynomial

$p(x) = x^3 + x + 1$. Usually LFSR starts generating the addresses with a seed value, for example, if the seed for the LFSR shown in figure 1.13 is 111, then the sequence of addresses generated will be {111,011,101,010,001,100,110.111}. A simple circuit can be added to generate the 000 address [28].

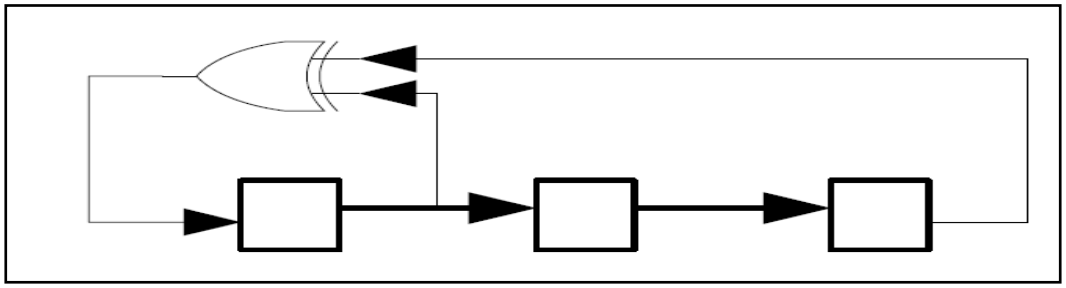

Figure 1.13: 3-bit LFSR

The controller in BIST engine is responsible for generating data and control signals that will be applied on the memory location generated by address generator. Usually the controller is based on finite state machine while generating its patterns [29].

BIST schemes are classified in two main types [28]:

1. Test per clock, in which BIST engine generates a pattern to be applied to the Circuit Under Testing (CUT) per clock.

2. Test per scan, in which the storage elements in the CUT are transformed into scan cells that are connected to each other forming a scan chain. Then, the testing vectors and responses are shifted through this chain. Usually this scheme is used for testing complex sequential circuits since it doesn't require large overhead area, but it results in large testing time.

Usually in memory testing, test per clock scheme is used due to its low testing time and there is no need for forming scan chains in testing memories since memory structure is the same for all memories.

1.4 Memory Testing Algorithms

Many algorithms were developed for testing memory. F**ault coverage**, which is defined as the number of detected faults divided by total number of faults, was considered as the superior factor in evaluating any memory testing algorithm. With

the increased number of embedded memories in the SoC, **testing time** became another important factor in testing. To reduce the testing time, parallel testing can be considered as a good solution, but this will result in excessive **test power** since multiple memory instances will be tested simultaneously. Hence, the power consumed during the testing mode could be much higher than that in the normal mode since in the testing mode multiple memories will be tested concurrently whereas some memories will be idle during the functional mode [30].

A memory testing algorithm consists of a sequence of read and write operations that will be applied on a sequence of addresses in the memory. In each read operation, the results read from MUT are compared with the expected read result to detect any faults in memory [31]. The following provides a brief description about the most commonly used memory testing algorithms. Figure 1.14 defines the symbols used in all memory testing algorithms:

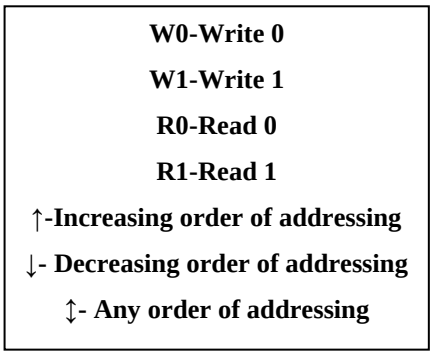

Figure 1.14: Symbols Used in Memory Testing Algorithms

1. **Zero-One algorithm:** This algorithm is used to detect only stuck-at faults in MUT. It is commonly used in personal and portable devices such as mobile phones. Also it is used when embedded memories of those devices are being tested during the application not only during manufacturing. As shown in figure 1.15, simply a 0 is written to all memory addresses, then it is read from those locations in order to detect stuck-at 1 cells. Then, 1 is written to all memory locations then it is read in order to detect stuck-at 0 cells. Note that the testing time of this algorithm is 4n where n is the number of addresses of the MUT. Thus, this algorithm belongs to O(n) testing algorithms [18]

↕ (W0), ↕ (R0), ↕ (W1), ↕ (R1),

Figure 1.15: Zero-One Algorithm

2. **March Tests:** It was proven that March tests are superior to other memory testing algorithms due to their high fault coverage and low testing time. In general, March algorithms are O(n) tests. They were found in order to detect coupling faults and other types of faults. A March test consists of a number of March elements, each element consists of a sequence of operations that will be applied on all memory addresses based on the given addressing order in the algorithm. Usually these tests are used for testing embedded memories in critical applications such as military applications and others so that the memory has to be intensively tested. One of the commonly used March tests is March C- algorithm which can detect stuck-at faults, transition faults, and inversion coupling faults. Figure 1.16 illustrates March C- algorithm and its main elements (M0, M1 …,M5) [32].

↕ (W0);↑ (R0 , W1); ↑ (R1 ,W0);↓(R0,W1); ↓(R1,W0); ↕ (R0)
M0 M1 M2 M3 M4 M5

Figure 1.16: March C- Algorithm

March C- algorithm contains increasing and decreasing orders of addresses in order to detect the coupling faults when the aggressor cell address is greater than the address of the victim cell (a > v) and when the aggressor cell address is lower than the victim cell address (a < v). Other March tests were developed in order to increase the fault coverage. For example, March C- algorithm cannot detect DRDFs, thus, March SS was found to detect this type and also other types of faults. Other March algorithms were found to detect and identify the location of the faults, such as March DSS. Those algorithms are called **diagnosis March Tests** [33]. More about March tests can be found in appendix A.

Zero-One and March algorithms are not the only algorithms developed for memory testing, other algorithms were found such as Walking 1/0 and GALPAT. Those algorithms have a good fault coverage but their complexity are $O(n^2)$ and $O(n^{3/2})$ respectively. Thus, a large testing time is needed [33].

Memory testing algorithms are implemented as a finite state machine which is executed by the controller in the BIST engine [29]. Figure 1.17 shows a finite state machine implementing the first two elements in March C- algorithm.

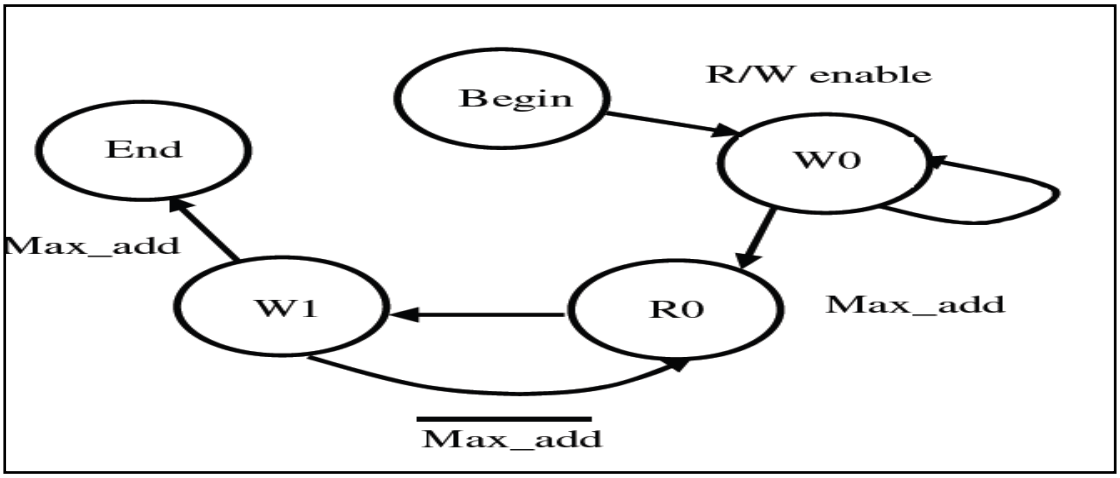

Figure 1.17: First two Elements of March C- [29]

1.5 Power Dissipation in SRAM Testing

Before considering power dissipation during SRAM testing, the following terminology has to be considered [28]:

1. Average Power: The total energy consumed divided by testing time.
2. Instantaneous Power: The power dissipated at any instant of time.
3. Peak Power: The maximum instantaneous power.

In CMOS technology, power dissipation can be classified in two types [30]:

1. Static power dissipation which is caused by leakage current. In general, leakage current increases when more and more transistors are squeezed onto a chip since when the transistor becomes smaller, the insulating layer becomes thinner causing more and more leakage current. In general, in CMOS technology, static power is low if compared with other technologies.
2. Dynamic power dissipation which is caused by charging and discharging of load capacitance (C_L) of the transistors. Figure 1.18 shows the general model of CMOS node. When a transition from low to high occurs, then the capacitor will be charged to Vdd with charge $Q= C_L Vdd$. So the energy consumed= $QVdd= C_L(Vdd)^2$. Since the load capacitor will save (1/2) $C_L(Vdd)^2$. The other half will be dissipated as

heat in the pull up network (P). In case of discharging, the load capacitor will discharge all the energy that is has and this energy cannot enter the ground rail since Q.Vdd=Q.0=0. Thus, (1/2) $C_L(Vdd)^2$ will be dissipated as heat in the pull down network (N). By this way, in any node, the dynamic power dissipated can be defined in (1.1).

$$\boldsymbol{Pdyn = ((1/2) * CL * Vdd^2 * N)/T} \qquad (\mathbf{1.1})$$

Where CL is the load capacitance of the transistor and Vdd is the biasing voltage, N is the total number of transitions in the node, and T is the testing time. Usually any transition is called **Switching Activity (SA)**. Note that power dissipation is the same for up and down transitions if assuming the same sizing of P and N transistors. [30].

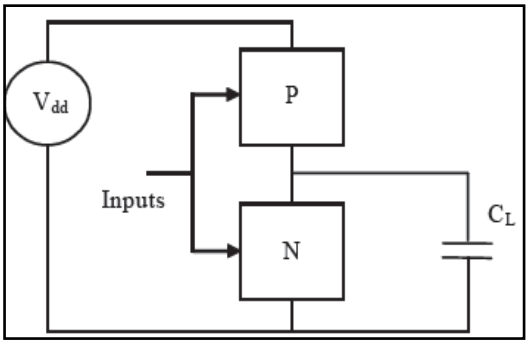

Figure 1.18: CMOS Logic [30]

Dynamic power is the dominant source of power dissipation when testing embedded SRAM. Thus, it is the power considered in this dissertation. Actually, dynamic power dissipation during SRAM testing is caused by three main factors [34]:

1. High Switching activities in address decoder and data bus.
2. Power dissipated in peripheral devices such as sense amplifier.
3. Power dissipated in memory array.

The power consumed in memory array forms the major part of power dissipation during testing. It is caused by read and write operations applied on memory cells. Write current is much higher than the read current since the value of the cell has to be overwritten with new value. Therefore, the voltage swing in write operation is set to Vdd. If write operation has to be performed on one cell in a word, the other cells in this word perform read operations whose results are neglected. These unnecessary read operations are called Read Equivalent Stress (RES).) Read and write drawn currents are represented in (1.2) and (1.3) respectively [35].

$$Idda(r) = [m * Idc(r) * Dt + m * Cd * DV(r)] * f \quad (1.2)$$

$$Idda(w) = \{p * [Idc(w) * Dt + Cd * DV(w)] + (m - p) * [Idc(r) * Dt + Cd * DV(r)]\} * f \quad (1.3)$$

Where m is word length, Idc(r) and Idc(w) represent the DC current on the bit lines during read and write operations respectively, Dt is the assertion period of WL, Cd is the capacitance of the bit line, p is the number of memory on which write operation will be applied, DV(r) and DV(w) represent the voltage swing of the read and write operations respectively, and f is the operation frequency. Note that in general m=p; however, for Bit/byte-Write Enable memories these parameters may be different. It is obvious that when a write operation is performed on a cell in a word, the other cells perform RES whose results will be neglected

1.6 Motivation for SRAM Test Power Reduction

SRAM testing entails excessive average power dissipation due to the large number of switching activities in address decoder and data bus in addition to the sequence of read and write operations applied during testing. Excessive average power means more heat which will increase the temperature of the chip; consequently, the chip may be damaged [30].

When parallel testing is performed on a number of embedded memories in SoC, there will be concurrent write operations that will result in excessive peak power. If peak power exceeds the power constraint, then the chip may be damaged. Excessive peak power will cause more noise that will erroneously change the logic value of nodes in the chip; hence, the system functionality will be impacted [30].

Although many algorithms were developed for SRAM testing, maximizing the fault coverage was the superior purpose of most of them, and only few of those algorithms focus on reducing the testing power which plays an important role in evaluating the effectiveness of the test. The limited number of techniques in reducing the testing power of SRAM was the main motivation of this dissertation.

1.7 Thesis Organization

This thesis aims to reduce testing power of Zero-One algorithm which is used for testing embedded SRAMs of personal devices, and to reduce the average and peak power of March tests which are used for intensive testing of embedded SRAMs used in critical applications. The last objective is to reduce the peak power when multiple embedded SRAMs in SoC are being tested in parallel.

In chapter 2, literature on previous work is summarized. This chapter will describe the previous algorithms developed for reducing SRAM testing power and how this dissertation contributes the previous work.

Chapter 3 of this thesis provides an enhancement on Zero-One algorithm to reduce the switching activity in the address decoder by using low power address generators, and reduces the switching activity in the data bus by reordering the algorithm. March tests, which are the most commonly used are modified in chapter 4 so that both the average and peak powers are reduced.

In chapter 5, a new scheme is proposed in order to reduce the peak power when large number of embedded SRAMs in SoC has to be tested in parallel. Finally, chapter 6 concludes this dissertation and describes the future work.

CHAPTER 2

Related Work

Due to the large number of faults in embedded SRAMs in SoC, many algorithms were developed in order to detect, and in some cases diagnose, these faults. Most of these algorithms focus on maximizing the fault coverage and reducing testing time. The increased complexity in VLSI technology results in more and more embedded memories in SoC, which has resulted in high power consumption in the chip .Testing power forms a major part of this power dissipation. Nevertheless, only few techniques were dedicated in order to reduce memory testing power.

The main motivation behind reducing testing power of embedded memories is that the power consumed during testing could be twice that in the functional mode since during testing multiple memories will be tested in parallel whereas some memories will be idle in the functional mode [30].

Some techniques in the literature focus on reducing the switching activity during testing in order to reduce the average power while others deal with reducing peak power when multiple embedded memories are being tested in parallel. Maintaining the same fault coverage and low overhead in the hardware area was a challenge in these techniques.

This chapter presents a short survey about previous works in reducing testing power of embedded memories and their shortcomings. Then it reveals the contribution of the work presented in this dissertation.

2.1 Single Bit Change (SBC) in Address Decoder

One of the main sources of power dissipation during testing is high switching activity in address bus since all memory locations have to be tested for faults. Thus,

reducing these signal activities will reduce testing power effectively. By this way, original memory testing algorithms have to be reordered so that the switching activity in address bus lines is minimized while retaining the same fault coverage. This is done by using SBC or gray code addressing to ensure that between two successive clock cycles, there will be only one transition. For example, if the MUT has 2 bit address bus, then the sequence of addresses generated during testing will be {00, 01, 11, 10} [30]. Table 2.1 shows how some memory testing algorithms can be modified in order to minimize the switching activity in address decoder. The symbol $\updownarrow_s$ denotes SBC in the addresses generated during testing.

Table 2.1: SBC in Address Decoder [30]

	Original Test	Low-power Test
Zero-One	$\updownarrow$ (W0); $\updownarrow$ (R0); $\updownarrow$ (W1); $\updownarrow$ (R1);	$\updownarrow_s$ (W0, R0, W1, R1);
Checker Board	$\updownarrow$ ($W(1_{odd}/0_{even})$); $\updownarrow$ ($R(1_{odd}/0_{even})$); $\updownarrow$ ($W(0_{odd}/1_{even})$); $\updownarrow$ ($R(0_{odd}/1_{even})$);	$\updownarrow_s$ ($W(1_{odd}/0_{even})$, $R(1_{odd}/0_{even})$, $W(0_{odd}/1_{even})$, $R(0_{odd}/1_{even})$);

It is clear in table 2.1 that Zero-One memory testing algorithm was modified so that the switching activity in address decoder is minimized. This is done by using SBC in the addresses generated and by applying all operations (W0, R0, W1, R1) on each address then moving to the next address instead of applying each operation to all addresses then re-generating all addresses to apply the next operation. Actually, original Zero-One test will cause generating the address sequence four times while they are generated only once in the low power version of this test.

The main drawback of SBC technique is that a modified counter is required in order to generate the required gray code. Actually using normal and gray code counters results in large overhead in the hardware area. In Zero-One testing algorithm, the order of addresses generated is not important, for this reason, it is preferred to use address generators with low hardware area instead of using counters since SoC contains a number of BIST engines and using counter as an address generator for each of those engines will make it costly for BIST in terms of hardware area.

2.2 Minimizing Test Power through Reduction of Pre-charge Activity

Before applying read and write operations in SRAM, the bit lines (BL, BLB) have to be pre-charged to Vdd to ensure applying a correct operation. The pre-charging circuitries are used for pre-charging and equalizing the high capacitive bit lines. It was proven that the pre-charging circuitry forms around 70% of the power consumed in SRAM [36]

A good methodology is to exploit the predictability of the sequence of addresses generated during testing in order to reduce testing power. Actually, during the functional mode, the next address that has to be accessed cannot be predicted whereas it is known during testing. Hence, all pre-charging circuits for all cells need to remain active in the normal mode whereas during testing mode, just the pre-charging circuits of the cells that have to be accessed will be activated while others can be deactivated [36].

Reducing the pre-charging activity during testing can be implemented by modifying the pre-charging control circuitry in a way that allows choosing a specific cell to be pre-charged. As shown in figure 2.1, a new element has to be added for each column for controlling the pre-charging circuit. This element consists of a multiplexer (which is implemented by two transistors and one inverter) and a NAND gate. The multiplexer has LP_{test} signal as a selection line in order allow selection between the normal mode and the testing mode. When LP_{test} signal is activated, then based on the addressing sequence (which is assumed to be word line after word line access), the signal CS_j' of column j drives the pre-charging circuitry of the next column j+1 while other cells in the same column are not pre-charged. NAND gate is used to allow the functional mode for a cell when it is selected for read and write operations during testing. Experimental results show that around 50% power reduction was achieved using this technique [36].

The main drawback of this technique is that it supposes word line after word line selection in addressing which is not used by all testing algorithms. .In many testing algorithms, read and write operations are applied in parallel to all cells within the same word in order to reduce the testing time. Thus, all pre-charging circuits have to be activated.

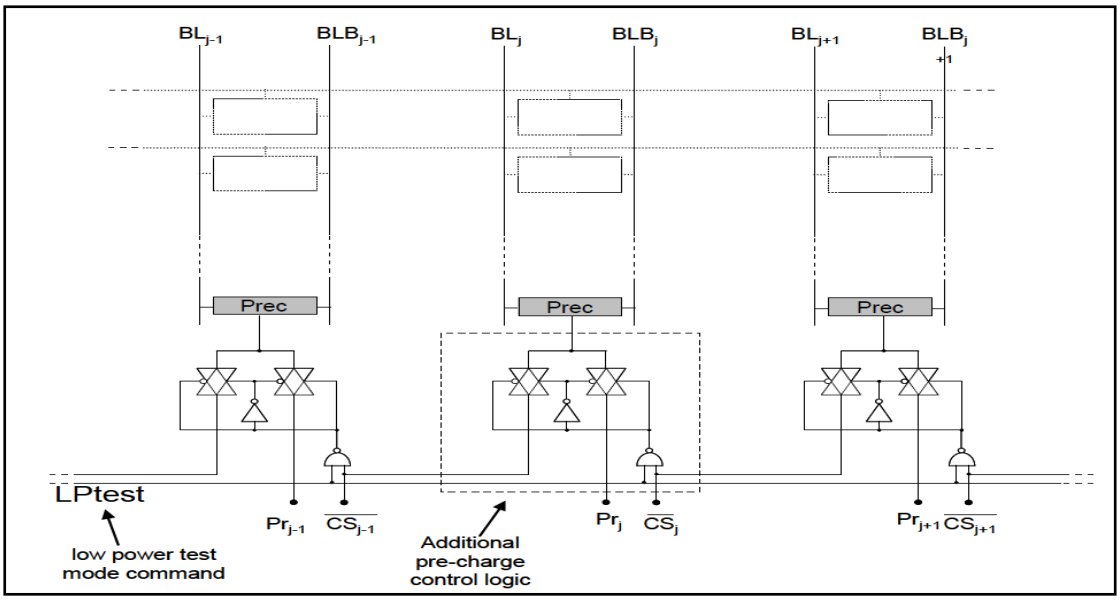

Figure 2.1: Modified Pre-charging Circuitry [36]

2.3 March Tests Sequence Reordering using Genetic Algorithm

Since March tests are the most commonly used for intensive testing of embedded memories, it is important to reduce their testing power. Switching activity during testing is the dominant source of power dissipation when applying March tests. This is related to the number of transitions during write operations. For example, in March C- algorithm, the number of transitions is 4 as shown in figure 2.3.

↕ (W0);↑ (R0 , W1); ↑ (R1 ,W0);↓(R0,W1); ↓(R1,W0); ↕ (R0)

Figure 2.2: Switching Activity in March C- Algorithm

One of the used techniques to reduce the switching activity in March tests is to reorder these tests based on genetic algorithm which is usually used for optimization. This algorithm can be used for optimizing between the fault coverage and testing power. For this reason, a cost function was defined based on the fault coverage and test power so that the fault coverage part is maximized, whereas the test power is minimized. The used cost function is represented in (2.1) [37] .

$$\boldsymbol{Cost(T) = W1 * (TP(T)/TPmin) + W2 * (FCmax/FC(T))} \qquad \textbf{(2.1)}$$

Where T is a March test that has a fault coverage FC(T) and test power TP(T), TPmin and FCmax represents the minimum test power and maximum fault coverage respectively. W1 and W2 represent the weights assigned to test power and fault

coverage. In general the cost for any March test T has to be minimized. The fitness function (F (T)) required for genetic algorithm is 1/Cost(T).

Genetic algorithm is based on starting with initial population, and then a number of genetic operations are applied to generate new populations. Periodically, the fitness function is calculated for each population generated so that population with the maximum fitness function is selected. The initial population for a given March test is the sequence of write operations, for example, the initial population for March C- test is the set {W0, W1, W0, W1, W0}. More about genetic algorithm can be found in appendix B.

This algorithm was applied on March B, March SS and March DSS (which is used for diagnosis) algorithms. Table 2.2 shows the old and newly generated low power March tests based on genetic algorithm.

Table 2.2: New Generated March Tests using Genetic Algorithm [37]

March B	{⇳(w0); ⇑(r0, w1, r1, w0, w1); ⇑ (r1, w0, w1); ⇓(r1, w0, w1, w0); ⇓(r0, w1, w0)}
March SS	{↕(w0); ↓(r0,r0,w0,r0,w1); ↑(r1,r1,w1,r1,w0); ↓(r0,r0,w0,r0,w1); ↓(r1,r1,w1,r1,w0); ↕(r0) }
March DSS	{ ⇑ (w0); ⇑ (r0, r0, w0, r0, w1); ⇑ (r1, w1, r1, w0); ⇓ (r0, w0, r0, w1); ⇓ (r1, w1, r1, w0, w0, r0); ⇓ (r0, w0, r0); ⇓ (r0, w1, r1);⇓ (r1, w1); ⇑ (r1, w0, r0); ⇑ (r0, w1, r1); ⇑ (r1, w0); ⇓ (r0, w0); ⇑ (r0, w1); ⇓ (r1, w1, w1);⇓(r1, w0); ⇓(r0)}
New March1	{⇳(w0); ⇑(r0, w0, r0, w0, w1); ⇑ (r1, w0, w0); ⇓(r0, w0, w0, w1); ⇓(r1, w1, w1)}
New March2	{⇳(w0); ⇑(r0, w0, r0, w0, w1); ⇑ (r1, w1, w1); ⇓(r1, w1, w1, w1); ⇓(r1, w0, w0)}
New March3	{↕(w0); ↓(r0,r0,w0,r0,w1); ↑(r1,r1,w1,r1,w0); ↓(r0,r0,w0,r0,w1); ↓(r1,r1,w1,r1,w1); ↕(r1) }
New March4	{ ⇑ (w0); ⇑ (r0, r0, w0, r0, w1); ⇑ (r1, w1, r1, w0); ⇓ (r0, w0, r0, w1); ⇓ (r1, w1, r1, w0, w0, r0); ⇓ (r0, w0, r0); ⇓ (r0, w0, r0);⇓ (r0, w0); ⇑ (r0, w0, r0); ⇑ (r0, w0, r0); ⇑ (r0, w0); ⇓ (r0, w0); ⇑ (r0, w1); ⇓ (r1, w1, w0);⇓(r0, w1); ⇓(r1) }

This technique reduces the testing power effectively in March test, but it performs this on individual memory whereas usually a large number of embedded memories have to be tested in parallel resulting in large peak power that may damage the chip.

2.4 Generating Low Power March Tests using Particle Swarm Optimization

This technique is similar to the one described in the previous section. It aims to generate low power March tests with low power and high fault coverage using Particle Swarm Optimization (PSO) scheme. Actually PSO is a population based stochastic scheme which is used to find a solution and then finding the optima through iterations. As in genetic algorithm, new populations are generated from some initial population by applying a number of operations. Then, the one with the maximum

fitness function is selected as optima. Usually, each particle consists of a sequence of read and write operations. Figure 2.4 illustrates the structure of PSO particle. The fitness function at kth generation of particle Pi equals to W1* (Fraction of faults covered) + W2*(Maximum power consumed by any particle till generation *k*)/ (Power consumed by the *March* test of particle *Pi* in generation *k*) +W3*(Maximum number of continuous writes in any particle till generation *k*)/ (Number of contiguous writes for the March test of particle *Pi* in generation *k*) [38].

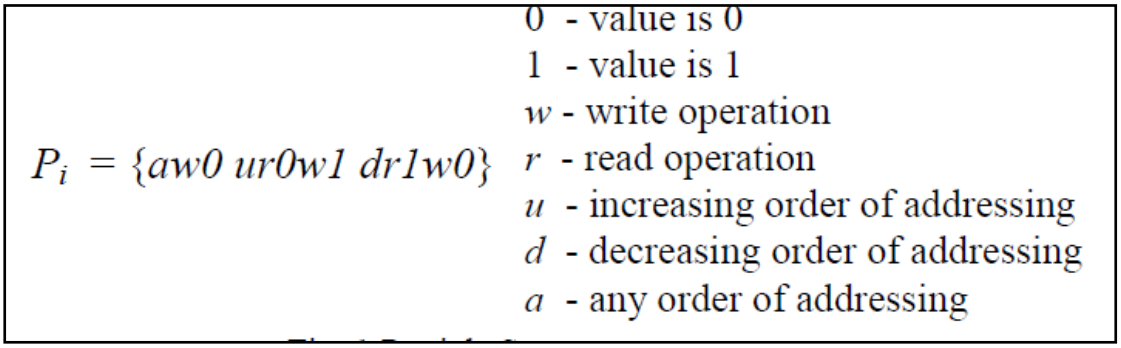

Figure 2.3: PSO Particle Structure [38]

The initial population in PSO scheme is generated by taking the Number Of Operations (NOP) of the March test as an input, then based on random variables, new particles are generated randomly. The fitness function is calculated for each of them. New particles are then generated using an operation called flip and the fitness function is calculated periodically. The particle with the best fitness function is selected as optima. More about PSO scheme can be found in appendix C.

Table 2.3 shows the newly generated March tests based on PSO scheme and their fault coverage and average power. It is obvious that this scheme is based mainly on the number of operations in the March test.

This algorithm reduces the testing power effectively through generating those low power March tests. But as in genetic algorithm, when a large number of embedded memories has to be tested simultaneously, then concurrent write operations will result in high peak power that may exceed the power constraint of the chip. Another drawback is that both this and genetic algorithm consider bit-oriented SRAM in their applications whereas most of the commonly used memories are word-oriented ones.

Table 2.3: New March Tests Based on PSO Scheme [38]

Algorithm	No P	FC	Power (mW)	W	Alternate Test Generated	FC	Power (mW)
March A	15	81.68	0.368	A	aw0 ur0w1r1 ur1w0 ur0 dr0w1 dr1w0 ur0 uw1r1 dr1	96.59	0.230
				B	aw0 ur0 dr0w1 ur1w0r0 dr0 uw1r1 dr1w0r0 ur0w1	95.01	0.230
March M	16	82.73	0.246	A	aw0 dr0w1 dr1 ur1w0r0 ur0w1 ur1w0w1 dr1 dr1w0 dr0	96.59	0.245
				B	aw0 ur0 dw1r1 dr1w0 ur0 ur0w1 ur1w0r0 dr0 dr0w1r1	96.16	0.219
March B	17	81.77	0.333	A	aw0 ur0w1 ur1 aw0 ur0w1 dr1w0r0 dr0w1 ur1w0w1w0 ur0	97.07	0.283
				B	aw0 dr0 dw1r1 ur1 dr1w0r0 dr0w1r1 ur1w0 ur0w1r1 ur1	93.57	0.210
March G	23	89.45	0.299	A	aw0 dr0w1w0r0 dr0w1 ur1w0r0w1 dr1w0 ur0w1 ur1w0r0 ur0 aw0 dr0w1 dr1	98.99	0.262
				B	aw0 ur0w1 ur1 aw0 dr0 dr0w1 dr1 aw0 ur0w1r1 ur1w0r0 dr0w1r1 dr1w0r0 dr0	96.93	0.226
ABL	37	94.72	0.252	A	aw0 ur0w1r1w0r0 ur0w1r1 uw0r0 dr0w1 dr1w0r0 ur0 uw1 dr1 dw0r0w1r1 ur1w0 uw1r1 ur1w0 ur0 uw1 dr1w0r0 dr0w1 ur1	98.56	0.243
				B	aw0 ur0w1r1w0r0 uw1r1 ur0w0 dr0w1r1 dr1w0r0 ur0 uw1 dr1 dw0r0w1r1 ur1w0 uw1r1 ur1w0 ur0 uw1 dr1w0r0 dr0w1 ur1	97.12	0.243

2.5 Skew Scheme

A SoC consists of a large number of embedded memories that have to be tested in parallel. When a memory testing algorithm, such as March C-, is applied on those memories, there will be concurrent write operations that will cause huge peak power that may damage the chip if it exceeds the power constraint. Figure 2.5 illustrates applying element M1 (R0,W1) in March C- algorithm on two SRAMs being tested in parallel. $(r0,w1)_0$ means that these operations will be applied on cell 0. So write operation will be applied on two cells concurrently. If the SoC consists of 100 memories, then 100 concurrent write operations will be applied during parallel testing which is dangerous to the cell due to the high current of write operation if compared with read operation.

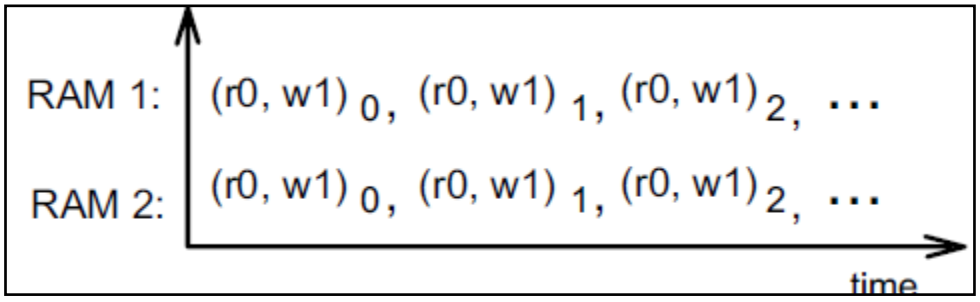

Figure 2.4: Applying element M1 of March C- on two SRAMs

The skew scheme was found in order to provide a good management for parallel testing of embedded memories. This is done by adding one clock cycle skew when testing two embedded memories to ensure that there will be no concurrent write operations. Figure 2.6 illustrates applying skew algorithm when element M1 of March

C- algorithm is applied on two SRAMs being tested in parallel. Let P(R) and P(W) denotes read power and write power respectively. In case of parallel testing without any skew, as shown in figure 2.5, peak power will 2*P(W) whereas it will be P(R)+P(W) when skew algorithm is applied as shown in figure 2.6. It is important to remember that always P(R) < P(W) [39].

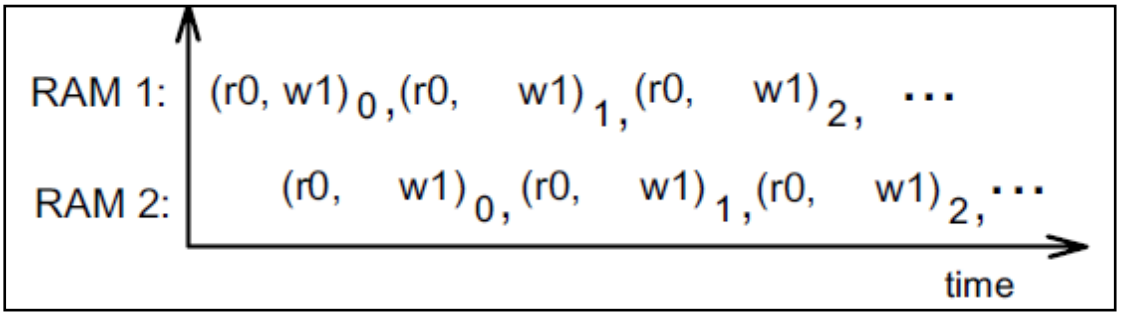

Figure 2.5: Skew Scheme [39]

In general, If N identical memories have to be tested in parallel, and grouped into two clusters such that the memories in each cluster will be tested in parallel, then, scheduling read and write operations as shown in Figure 2.6 will result in a peak power reduction from N*P(W) to (N/2)* (P(W)+P(R)).

The main advantage of skew scheme is that it reduces the peak power effectively with just one additional clock cycle in the testing time. Nevertheless, if the power constraint of the chip is low, then skew scheme may be insufficient in reducing peak power since the memories in each cluster will be tested in parallel and may exceed this constraint.

2.6 Power Constrained Embedded Memory BIST Architecture

A good architecture was developed for MBIST in order to reduce the testing power and routing in connections. As shown in figure 2.7, this architecture consists of three main parts [40]:

1. MBIST controller in which the test program is stored and it starts the test after receiving a special command from upper level controller.

2. Low power MBIST wrapper which can be considered as the test pattern generator for the MUT and it is triggered by the controller to apply patterns.

3. Interconnections between controller and wrappers which is serial in order to reduce the routing complexity.

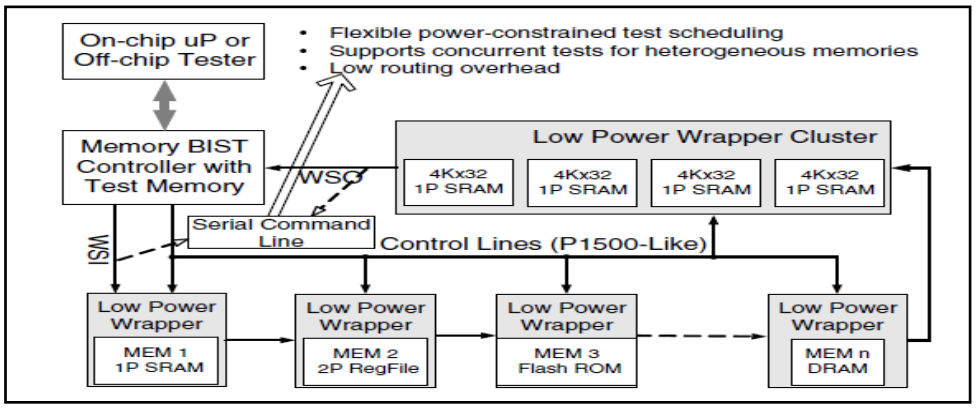

Figure 2.6: Power-Constrained MBIST Architecture [40]

Usually, MBIST controller sends commands to different wrappers that start applying testing patterns on the memories that have to be tested. Each wrapper uses gray code address generator in order to reduce the signal activities during testing. This generator can be up or down gray code counter. Each wrapper can be used to apply patterns on a single memory. Figure 2.8 illustrates the structure of address generator used in wrapper [40].

This architecture doesn't provide a management of parallel testing, since although the switching activity is reduced, parallel testing will result in abrupt increase in the peak power.

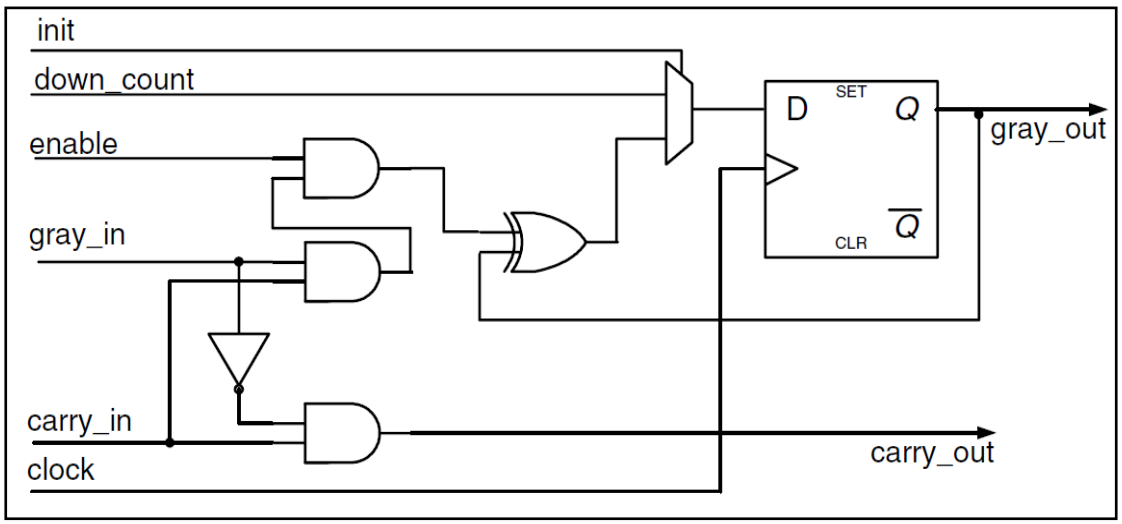

Figure 2.7: Wrapper Address Generator [40]

2.7 Contribution of the Work Presented in this Thesis

This dissertation introduces a number of techniques for reducing testing power of embedded SRAMs in SoC. These techniques aim to enhance some of the algorithms described in this chapter to overcome their drawbacks.

In chapter 3, low power and low hardware area address generator is suggested in order to be used for Zero-One algorithm which is used for testing personal devices. So the switching activity in the address decoder is reduced with little overhead in hardware area. Then, the test is reordered so that the write drivers switching activity is reduced.

In chapter 4, an enhancement is applied on March tests for word oriented memories in order to reduce their peak and average power. Finally, a management of parallel testing of embedded memories is proposed in chapter 5. This is done by applying a new scheme which improves the skew scheme effectively.

The work presented in chapters 4 and 5 was accepted for publication in the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT 2011) that will be held in 3-5 October, 2011 in Vancouver, Canada. Another paper about the contribution presented in chapter 3 is prepared to be submitted for another conference.

CHAPTER 3

Low Power Zero-One Testing

3.1 Overview

Stuck- at faults can be considered as the most common types of faults in SRAM cells. Usually, embedded memories of personal and mobile devices such as mobile phones and digital cameras are tested mainly for this type of faults. Actually, this is done during manufacturing and even in on-line testing, in which memory is tested when it is used in the application [18].

As mentioned in chapter 1, Zero-One algorithm is used for detecting stuck-at faults. This algorithm needs to be enhanced in order to reduce the power consumed during testing since power is a crucial factor for mobile devices especially if on-line testing is being applied.

This chapter introduces an enhancement of Zero-One algorithm in order to reduce the power consumed during testing. This enhancement consists of two main parts:

1. Selecting a low power address generator to reduce the switching activity in address decoder.

2. Reordering the Zero-One pattern so that total switching activity is minimized.

Section 3.2 describes the used MBIST architecture and address generators. Section 3.3 reorders Zero-One testing pattern so that total switching activity is minimized. Section 3.4 reports and analyzes obtained results while section 3.5 concludes this chapter.

3.2 MBIST Architecture and Address Generators

In MBIST, the BIST engine generates a number of testing vectors that are applied on MUT. Each vector consists of address that has to be accessed, data background that has to be written, and control signal to determine whether a read or a write operation has to be applied. Figure 3.1 shows the design for MBIST used in this chapter. Basically, the BIST engine (controller) provides mr and mw signal for read and write operations respectively. It also generates the address that has to be accessed and the data that will be written to MUT in case of write operation. If the operation is read, then the expected data to be read will be sent to the comparator that compares between this data with the output of MUT after read operation. The Mux has the test signal as its selection line in order to determine whether the system is in the functional mode or in the testing mode. In general, if the system is in the testing mode, then the patterns sent by BIST controller will be applied to the memory while the data coming from microprocessor will be applied if the system is in the normal mode. Finally, the comparator compares in parallel the data read from memory with the expected data; if these values are different, then the test fail signal will be activated indicating that the MUT is faulty. The test continues until the test end signal is activated by the BIST controller. More about building MBIST systems can be found in [41].

One of the possible enhancements for Zero-One pattern is to exploit the fact that during testing, the order of addresses generated is not important. For this reason, a low power and lower hardware area address generator has to be used. Since using a counter entails a large overhead in the hardware area, LFSR is usually used for this purpose [28]. As described in section 1.3, with a little hardware area and with the appropriate selection of XOR gate location, LFSR can generate all the possible addresses except the zero's address which can be generated using a simple circuit. This will not affect the fault coverage since all locations will be generated and tested.

Although LFSR occupies a low hardware area, there is a low correlation between the addresses generated. This causes a high switching activity in the address decoder which increases the heat dissipated during testing. Thus, other types of LFSR were developed in order to reduce the switching activity in the vectors generated. These LFSRs are used for testing combinational and sequential circuits, but they were not used for testing memories. Hence, these LFSRs were implemented, simulated, and then compared with each others in terms of their switching activity in order to find the

best one to be used as an address generator when testing SRAMs for stuck-at faults. The following sections provide a brief description about these types of LFSR.

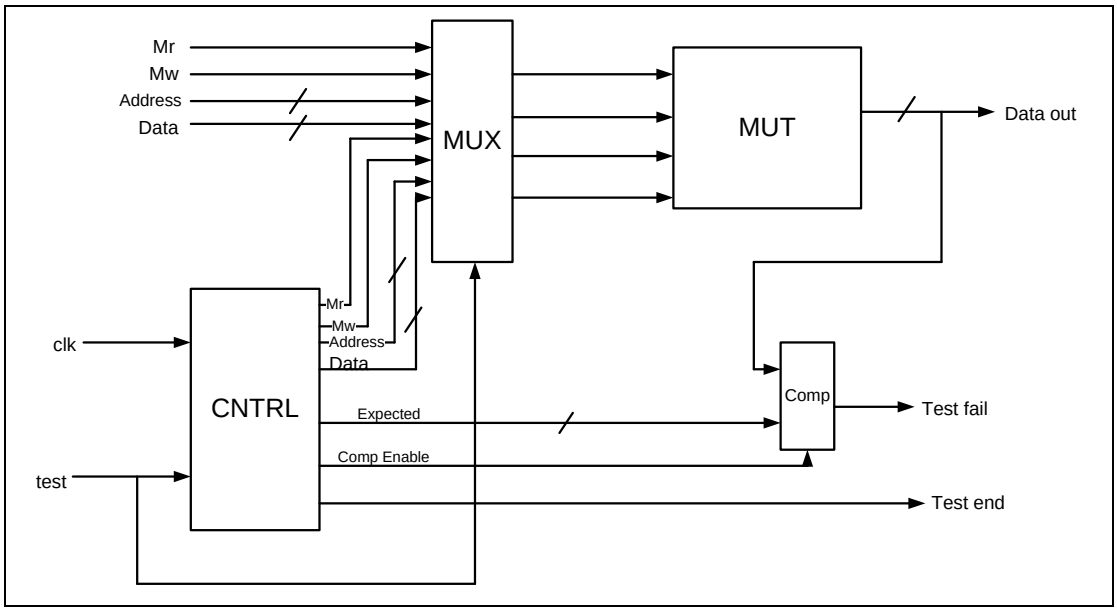

Figure 3.1: Used MBIST Architecture

3.2.1 Bit-Swapping LFSR (BS-LFSR)

In this type, the LFSR structure is modified to apply swapping between the neighboring bits. The last bit is the selection line for the swapping process. If the last bit is 0, then swapping is performed between neighboring flip flops, otherwise, nothing is changed. As shown in figure 3.2, just a number of multiplexers have to be used to allow swapping. For example, if a 3-bit LFSR has to be used, then the generated vectors using normal LFSR (such as the one shown in figure 1.11) and BS-LFSR are shown in table 3.1. So each vector generated by normal LFSR is checked to see whether it has to be swapped or not. It was proven that BS-LFSR reduces the switching activity in the inputs of the CUT about 25% [42].

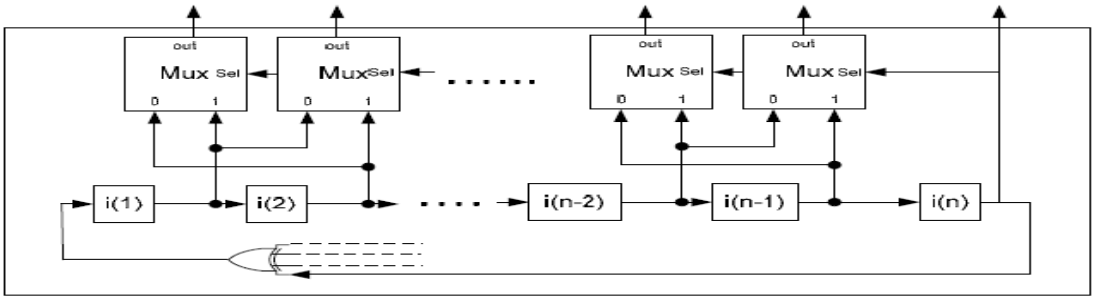

Figure 3.2: BS-LFSR [42]

Table 3.1: Normal and BS- LFSR Vectors

Original Vector	Swapped Vector
111	111
011	011
101	101
010	**101**
001	001
100	**010**
110	**110**
111	111

3.2.2 Dual Speed LFSR (DS-LFSR)

This LFSR is commonly used in testing since it reduces the switching activity effectively. Instead of using one LFSR, two LFSRs are used: slow speed LFSR and normal speed LFSR. The slow-speed LFSR is driven by a slow clock whose speed is a fraction of the clock that drives the normal-speed LFSR. Figure 3.3 illustrate this type of LFSR. When the normal-speed LFSR finishes all its vectors, the slow-speed LFSR clock is triggered [43].

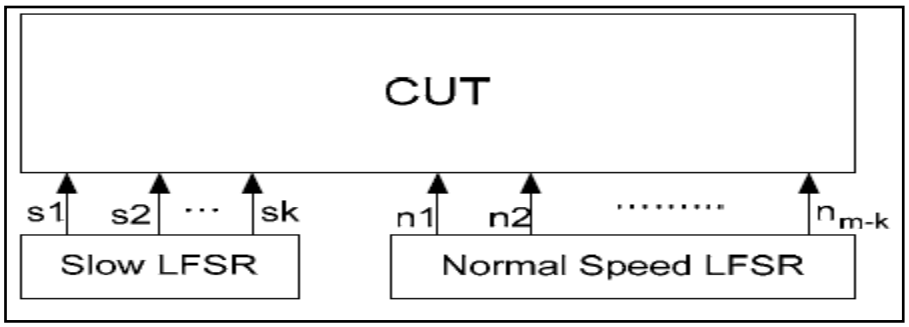

Figure 3.3: DS-LFSR [43]

The main feature of DS-LFSR is that it reduces the frequency of transitions in the circuit inputs that are driven by the slow-speed LFSR. So the total number of switching activities is reduced.

3.2.3 Bipartite LFSR

This LFSR is based on reducing the switching activity between two consecutive patterns through combining the second half of the current vector with the first half of the next vector into an intermediate vector. As shown in figure 3.4, the switching activity is divided into two stages [44].

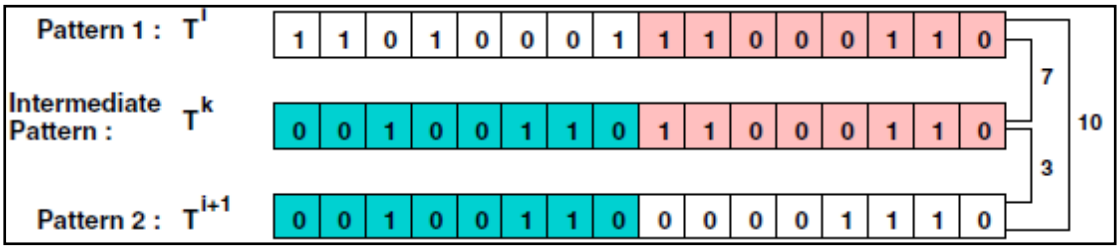

Figure 3.4: Intermediate Pattern Generation in Bipartite LFSR [44]

Bipartite LFSR can be implemented simply by dividing the LFSR into two halves so that when one half is working, the other half is idle. Actually two enable signals are required (en1, en2) as shown in figure 3.5. When en1en2=10, then the first half is working while when en1en2=01, then the second half is working. An intermediate flip flop is added to store the value of n/2th flip flop when the first half is active and sends it value to (n/2+1)th flip flop when the second half becomes active [44].

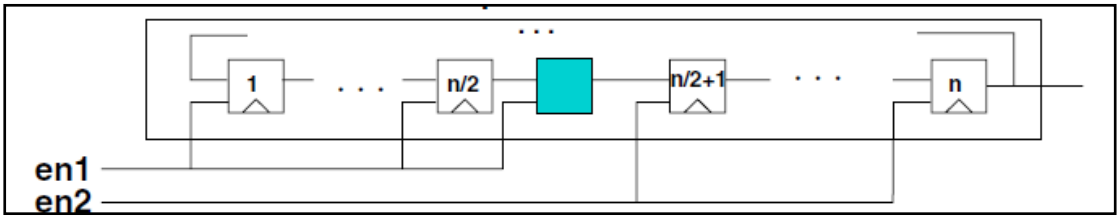

Figure 3.5: Bipartite LFSR Architecture [44]

Bipartite LFSR reduces instantaneous power, but for average power, it will be the same if all vectors are generated since it only divides the switching activity in two parts (for example from 10 to 7 and 3 as shown in figure 3.4). Another disadvantage is that if this generator was used for the same testing time of normal LFSR, then some addresses will be duplicated while others will not be generated, and this will affect the fault coverage of the test.

3.3 Detecting Stuck-at Fault Patterns

In order to detect stuck-at faults, Zero-One pattern writes zero to all memory locations, then it reads zero from all these locations so that stuck-at one cells are detected. After that, one is written to all cells and then read to detect stuck-at zero cells. As mentioned in section 2.1, this pattern causes a high switching activity in the address decoder since all addresses will be generated four times and in case LFSR is being used, there will be huge switching activity, thus, the technique proposed in section 2.1 suggests to modify Zero-One pattern from {↕(W0); ↕(R0); ↕(W1); ↕(R1)} to {↕(W0,R0,W1,R1)}, so that all operations are applied on one cell then moving to the next cell. This will reduce the switching activity in the address decoder since the addresses will be generated just one time. To make it easier, the original Zero-One pattern will be called **pattern 1** while the later one will be called **pattern 2.** Switching activity of pattern 1 (SA_{p1}) can be represented in (3.1)

$$\boldsymbol{SA_{p1} = 4 * SA_{add} + p} \qquad \textbf{(3.1)}$$

Where SA_{add} is the address decoder switching activity and p is the word length of the MUT. Although pattern 2 reduces the switching activity in the address decoder, it entails a problem in the switching activity in the data bus since most of the memories used in reality are word oriented memories. For example, if the word length of the MUT is 8bits, then using pattern 2 will result in huge switching activity in the data bus since when each location has to be accessed, such as the second location, the data bus will have 16 transitions (11111111→00000000→11111111). Higher word lengths will result in higher switching activities. Switching activity of pattern 2 (SA_{p2}) is represented by (3.2)

$$\boldsymbol{SA_{p2} = SA_{add} + p * (2N - 1)} \qquad \textbf{(3.2)}$$

Where N is the number of memory locations (words) in the MUT. Note that in each word, there will be 2p transitions from ones to zeros and then from zeros to ones except the first location which contains p transitions from zeros to ones (if it is initially assumed to contain zeros). It is obvious that pattern 2 has a bad impact on data bus switching activity since it is proportional to the word length and the number of locations in the MUT.

. To reduce switching activity, there should be some optimization between the address bus switching activity and data bus switching activity. First of all, the lowest power address generator has to be selected, and then, a suitable pattern has to be

applied so that there is no excessive power consumed either in the address decoder or in the data bus. By this way, Zero-One pattern can be modified to be {↕(W0,R0); ↕(W1,R1)} and let this pattern called **pattern 3.** So when applying this pattern, the data bus will contain one switching activity in each of its lines whereas the test goes through the addresses two times. Switching activity of pattern 3 (SA_{p3}) is represented by (3.3).

$$\boldsymbol{SA_{p3} = 2 * SA_{add} + p} \qquad \textbf{(3.3)}$$

It is clear that pattern 3 is better than pattern 1 since the addresses are generated two times; also it is better than pattern 2 since it doesn't have a linear relationship with the word length of MUT. Thus, using pattern 3 with low power address generator will reduce the testing power of Zero-One pattern effectively. Note that the fault coverage is the same for all patterns since all addresses will be generated and tested.

3.4 Simulations and Experimental Results

To evaluate the proposed patterns and address generators, MBIST architecture shown in figure 3.1 was programmed using VHDL, then it was simulated using Xilinx ISE Design Suite [45].This tool is very powerful in simulations and it contains a power analyzer. Five address generators were used in the system, and then the switching activity was calculated in the address decoder when using each of these generators. Thereafter, patterns 1, 2 and 3 were programmed and total switching activity was calculated in order to select the best pattern with the best address generator.

3.4.1 Code Description

To build the used MBIST system, behavioral model with VHDL was used to build each of the components in the system and integrate them together. Four components were build and then they were connected with each other using port map, these components are:

1. TPG that generates the pattern that will be applied to the MUT. This component consists of two sub components: address generator that generates the address and the controller. The controller is a finite state machine that implements the testing algorithm; it triggers other

components such as the address generator and comparator. For example, if pattern 2 has to be implemented, it triggers the address generator, and then sends four operations (W0, R0, W1, and R1). In each read operation it activates the comparator enable signal and sends the expected data to be read to the comparator. When the test is finished, it activates test end signal.

2. Multiplexer that has the test signal (which is input in the system and set to 1 before start simulation) as its selection line. When the test signal is activated, the output of this component is the pattern sent from the TPG.
3. MUT which is implemented as 1D*1D array. If the control signal is read, then the value stored in the address is the output of the memory while the input data is written to the address in case of write operation.
4. Comparator that compares between the output of the memory and the expected data sent from the TPG. If the values are different, the test fail signal is activated.

After building the system, different address generators can be used by implementing each of them in the address generator component. Different patterns can be implemented by modifying the controller component such as the time to trigger the address generators and the sequence of operations. To increase the usability of the code, a package was used to set input values, such as memory size, seed of LFSR and others. Reset signal for LFSR has to be triggered in the beginning of the test.

3.4.2 MBIST Simulation

Figure 3.6 shows the simulation of MBIST system shown in figure 3.1. In this figure, a fault free memory was simulated with a 3-bit normal LFSR as an address generator. Pattern 2 was used so that all read and write operations are applied on one memory address before proceeding to the next address. After simulation, **SAIF** (Switching Activity Interoperation Function) command in the simulator was used to generate XML file that contains the switching activity in each signal in the system.

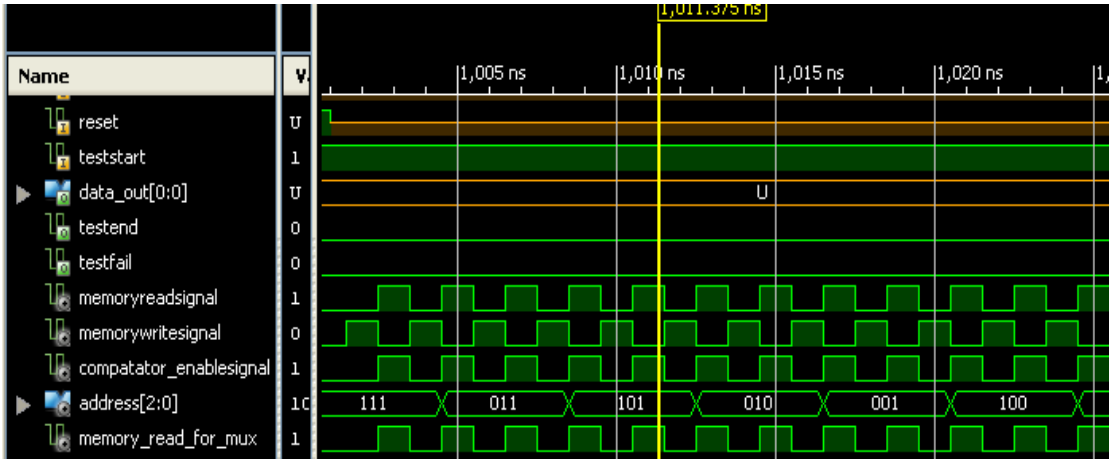

Figure 3.6: Fault Free Memory Simulation

3.4.3 Address Generators Simulations and Results

To find address generator with the least switching activity, five maximal length address generators were used in the system and compared in their switching activities. These generators are: LFSR, BS-LFSR, DS-LFSR, bipartite LFSR, and DS-LFSR with BS-LFSR for its slow and normal generators. The generators were programmed so that the zero address is generated after generating all other addresses. Tables 3.2 and 3.3 show the switching activity in each of these address generators and the saving percentage for each of them if compared with normal LFSR. Two seeds were used: the first one is "111…11" whose results are in table 3.2 while the other seed is "010101….01" whose results are in table 3.3. The switching activity shown in the tables represents the number of transitions (from high to low and from low to high) which is proportional to dynamic power dissipation as proven in section 1.6. More about maximal length LFSR can be found in appendix D.

It is obvious that normal LFSR causes high switching activity in the address decoder due to the low correlation in the addresses generated since it is a pseudo random generator. The main advantage of this generator is its low overhead in the hardware area. BS-LFSR reduces the switching activity effectively if compared with normal LFSR due to swapping process of neighboring flip flops. BS-LFSR generates all the addresses that the normal LFSR generates, also the switching activity is the same even if the seed changes since BS-LFSR is based on generating addresses using normal LFSR and then applying swapping process.

Table 3.2: Switching Activity for Address Generators with Seed "1111….1"

Address Bus Length (bits)	Testing Time (ns)	LFSR (SA)	BS-LFSR (SA)	Saving %	DS-LFSR (SA)	Saving %	Bipartite LFSR (SA)	Saving %	BS & DS LFSR (SA)	Saving%
5	128	85	69	19	70	18	45	47	64	25
10	4096	5130	4106	20	2904	43	2462	52	2376	54
15	131072	245775	188431	23	133324	46	121659	50	108652	56
20	4194304	10485780	8126484	22	5266450	50	5238247	50	4216850	60
25	134217688	419430211	318767107	24	218202082	48	209716053	50	167889901	60
30	152976488	516276423	454130346	12	286869408	44	286733465	44	219941228	57

Table 3.3: Switching Activity for Address Generators with Seed "0101…..01"

Address Bus Length (bits)	Testing Time (ns)	LFSR (SA)	BS-LFSR (SA)	Saving %	DS-LFSR (SA)	Saving %	Bipartite LFSR (SA)	Saving %	BS & DS LFSR (SA)	Saving%
5	128	85	69	19	55	35	48	44	57	33
10	4096	5130	4106	20	2643	48	2428	53	2315	55
15	131072	245775	188431	23	131524	46	121693	50	107626	56
20	4194304	10485780	8126484	22	5250055	50	5239003	50	4199893	60
25	134217688	419430211	318767107	24	218128300	48	209749569	50	167881672	60
30	152976488	516276423	454130346	12	286469208	44	277195287	46	219921231	57

It could be found that DS-LFSR is more efficient in reducing the switching activity than the BS-LFSR. Actually with large address spaces DS-LFSR is preferred to be used since the frequency of transitions is reduced in the lines connected to the slow speed LFSR. Nevertheless, this generator requires synchronization between the slow and normal clocks.

Bipartite LFSR has low switching activity as shown in the results. The main problem related to this generator is that it reduces the instantaneous power not the average power. Some addresses may be redundant since it generates intermediate vectors that may appear more than one time in the sequence. To cover all memory locations, more testing time is required and in this case the total switching activity will be the same as normal LFSR.

DS-LFSR with BS-LFSR for its low and normal generators has the least switching activity in address decoder among other generators since swapping process is applied for both slow and normal generators outputs and also the frequency of transitions is reduced in the lines connected to the slow speed generator. This

combination has also low overhead in the hardware area. For this reason, this generator is recommended to be used when SRAM has to be tested for stuck-at faults.

3.4.4 Testing Patterns Simulations and Results

The three patterns (1, 2, 3) described in section 3.3 were applied when simulating the system. Table 3.4 shows the switching activity in addresses bus and data bus when applying these patterns. In this simulation, DS-LFSR with BS-LFSR for its slow and normal parts was used as an address generator since it has the least switching activity in the address decoder. Word oriented SRAM was considered in this test with 16-bit for each memory location.

Table 3.4: Address and Data Bus Switching Activities for Different Patterns

Address Bus Length (Bit)	Pattern 1		Pattern 2		Pattern 3	
	Address Bus (SA)	Data Bus (SA)	Address Bus (SA)	Data Bus (SA)	Address Bus (SA)	Data Bus (SA)
5	256	16	64	992	128	16
10	9504	16	2376	32736	4752	16
15	434608	16	108652	1048544	217304	16
20	16867400	16	4216850	33554400	8433700	16
25	671559604	16	167889901	1073741712	335779802	16
30	879764912	16	219941228	5040920480	439882456	16

Table 3.4 clearly shows that pattern 1 is the worst since it causes huge switching activity in the address decoder since it generates the address sequence four times. The second pattern is more efficient than the first one if bit oriented memory was considered, but most of the memories used in reality are word oriented ones. Therefore, this pattern will cause huge switching activity in the data bus since the number of transitions increases with the size of MUT.

Pattern 3 could be considered as the best one among others. The address decoder switching activity is accepted since the sequence of addresses is generated two times and it is low for data bus since the data background changes only one time during the test. Hence, this pattern is recommended to be used when testing word oriented SRAMs for stuck-at faults. This pattern is applicable for all sizes of memories, either the ones with small address space and wide word lengths or the ones with small word length and large address spaces.

3.5 Summary

In this chapter, a number of techniques were proposed in order to reduce power consumption when SRAM has to be tested for stuck-at faults. Results show that using a combination of BS and DS LFSRs will result in low switching activity in the address decoder and in low overhead in the hardware area. The testing pattern plays an important role in power dissipation. When choosing a testing pattern, the switching activity in both address and data buses should be taken into consideration. It was proven that optimal switching activity can be obtained by using the pattern {↕(W0,R0); ↕(W1,R1)} with DS-LFSR that has slow and normal BS-LFSRs as an address generator. Fault coverage will not be affected since all addresses will be generated and tested for stuck-at faults.

CHAPTER 4

Low Power March Tests

4.1 Overview

When SRAM has to be intensively tested, several types of tests have to be applied on it during the testing phase after manufacturing. In each test, several types of faults have to be detected within an accepted testing time. Maximizing the fault coverage and reducing the testing time was considered as the main purpose for any testing algorithm.

Due to their high fault coverage and accepted testing time, March tests with complexity O(n) were considered as the superior for other tests. Many March tests were developed in order to detect more faults such as March C- which was found to detect coupling faults, March SS that detects more faults such as WDFs, and DRDFs, March DSS which is used for fault diagnosis. [46].

Since March tests are widely used during manufacturing, testing power has to be reduced. As described in chapter 2, several algorithms were used in order to generate new March tests with low power such as genetic algorithm and PSO scheme. Nevertheless, these algorithms consider bit oriented memories in their tests whereas most of the memories used in reality are word oriented ones.

In this chapter, March tests are modified so that peak and average power during testing word oriented memories are reduced. Since it is commonly used, March C- algorithm will be considered and modified based on the proposed algorithm which is applicable to be used with other March tests. March C- algorithm is shown in figure 4.1

Section 4.2 proposes the algorithm that will be used in reducing power of March C- algorithm. Section 4.3, shows how this algorithm can be implemented. Section 4.4 deals with the fault coverage of modified March C- and how it could to be maximized. Section 4.5 reports experimental results and section 4.6 concludes this chapter.

↕ (W0);↑ (R0 , W1); ↑ (R1 ,W0);↓(R0,W1); ↓(R1,W0); ↕ (R0)

Figure 4.1: March C- Algorithm

4.2 Modified March C- Algorithm

March C- algorithm consists of a sequence of write and read operations that are applied to MUT. As described in section 1.5, write operation consumes much more current than read operation to override the current value of the cell. Usually when a word-oriented memory is being tested, the data background (i.e., the data that will be written to MUT which belongs to {0,1}) is written to all bits in the word. Thus, in equation (1.5), the values of m and p will be the same, so the write current (Idda(w)) equation will be:

$$\boldsymbol{Idda(w) = p * \{Idc(w) * Dt + Cd * DV(w)\} * f} \qquad \textbf{(4.1)}$$

Where p is the word length of MUT, Idc(w) is the DC current on the bit lines during write operation, Dt is the assertion period of WL, Cd is the capacitance of bit lines, DV(w) is the voltage swing during write operation and f is the operation frequency.

(4.1) clearly shows that the write current is proportional to the word length of the MUT. For example, if the word length is 512 bit, then write operation will be applied to all these bits resulting in high peak power that may damage the MUT.

One of the possible solutions to reduce peak power of a March test is to divide the word of MUT into two clusters so that write operation is applied to one of these clusters. By this way, the power consumed during write operation will be reduced since the number of cells that will be written simultaneously is reduced. So write operation will be performed on the cells of the first cluster while cells of the second cluster will perform RES operations whose values will be neglected.

This idea can be applied on March C- algorithm by dividing SRAM into two clusters so that even cells belong to one cluster whereas odd cells belong to the other

cluster. When write operation is applied on one cluster, cells of the other cluster are disconnected from their data bus. Figure 4.2 illustrates **Modified March C- Algorithm.**

↕(W0x,Wx0);↑(R0,Wx1);↑(R01,W1x);↓(R1,Wx0); ↓(R10,W0x);↕(R0)

Figure 4.2: Modified March C- Algorithm

W0x means zero will be written to all even cells in the word (0,2,4,….) whereas odd cells will be disconnected from the data bus. On the other hand, Wx0 means zero will be written to all odd cells (1,3,5,…). So the algorithm starts with writing 0 to all memory words; in each word, write operation is applied first to even cells, then it is applied to odd cells. After that, 0 is read from all locations and 1 is written to all odd cells in each word. In the third element, 01 is read from all locations and 1 is written to all even cells. With decreased order of addresses, the fourth element reads 1 from all locations and then writes 0 to all odd cells. 10 is read from all locations during the fifth element and 0 is written to all even cells. Finally, read 0 operation is applied to all cells.

When applying original March C- algorithm on memory, the peak power ($Peak_{original}$) can be represented in (4.2):

$$\boldsymbol{Peak_{original} = P(W) * p} \quad \textbf{(4.2)}$$

Where P(W) represents the power of write operation and p is the word length of MUT. When modified March C algorithm will be applied, the peak power ($Peak_{modified}$) can be represented in (4.3)

$$\boldsymbol{Peak_{modified} = (p/2) * (P(W) + P(R))} \quad \textbf{(4.3)}$$

Where P(R) represents the power of read operation. Note that when write operation is applied on one cluster, the cells of other cluster perform read operations. Always P(R) is less than P(W) and theoretically if P(R) << P(W), then 50% reduction can be achieved in peak power.

Average power is proportional to the number of operations applied during testing. Average power can be defined as the total power consumed due to read and write operations divided by the total number of operations. For original March C- algorithm, the average power ($Avg_{original}$), when considering one word can be represented in (4.4). It is important to note that March C- algorithm consists of 5 read operations and other 5 write operations.

$$\boldsymbol{Avg_{original} = (5p * [P(R) + P(W)])/10} \quad \textbf{(4.4)}$$

Where p is the word length of MUT and 10 is the total number of operations in March C- algorithm. Each operation will be applied to all cells in the word**,** thus, in (4.4), each operation is multiplied by p. If modified March C- algorithm is applied on one word, the average power ($Avg_{modified}$) can be represented in (4.5). Note that modified March C- algorithm consists of 6 write operations and 5 read operations.

$$\boldsymbol{Avg_{modified} = (5p * P(R) + 6 * (p/2) * [P(W) + P(R)])/11} \quad \textbf{(4.5)}$$

Average power of the modified March C- algorithm consists of three parts: the first part is the power consumed during read operations which is applied to all cells in the word. The second part is the power consumed during write operations which is applied to just one cluster at any moment of time. The last part is the read equivalent stress operations that are applied to the idle cluster during write operation. Since usually P(R) <P(W), the weight of write power will reduce effectively since this operation will be applied to half of the cells in the word. If P(R) << P(W), then average power reduction will be significant.

4.3 Implementation

To implement Modified March C-algorithm, tri-state buffer could be used in order to disconnect unwanted cells during the write operation. Figure 4.3 illustrates the connections between the data bus and 4-bit word. If the mode signal (M) is 0, then the data background will be written to even cells (0, 2), whereas, it will be written to odd cells (1, 3) when mode signal is 1. MComp is the complement of signal M. D0, D1, D2 and D3 are the data bus lines.

4.4 Fault Coverage

March C- algorithm was found mainly in order to detect coupling faults in addition to transition and stuck-at faults. As shown in figure 4.1, this algorithm contains two modes in moving from one address to the next one: Increasing (↑) and decreasing (↓)**.** Using two orders is necessary in order to detect coupling faults in two cases:

1. When the aggressor cell address is higher than the victim cell address.
2. When the aggressor cell address is lower than the victim cell address.

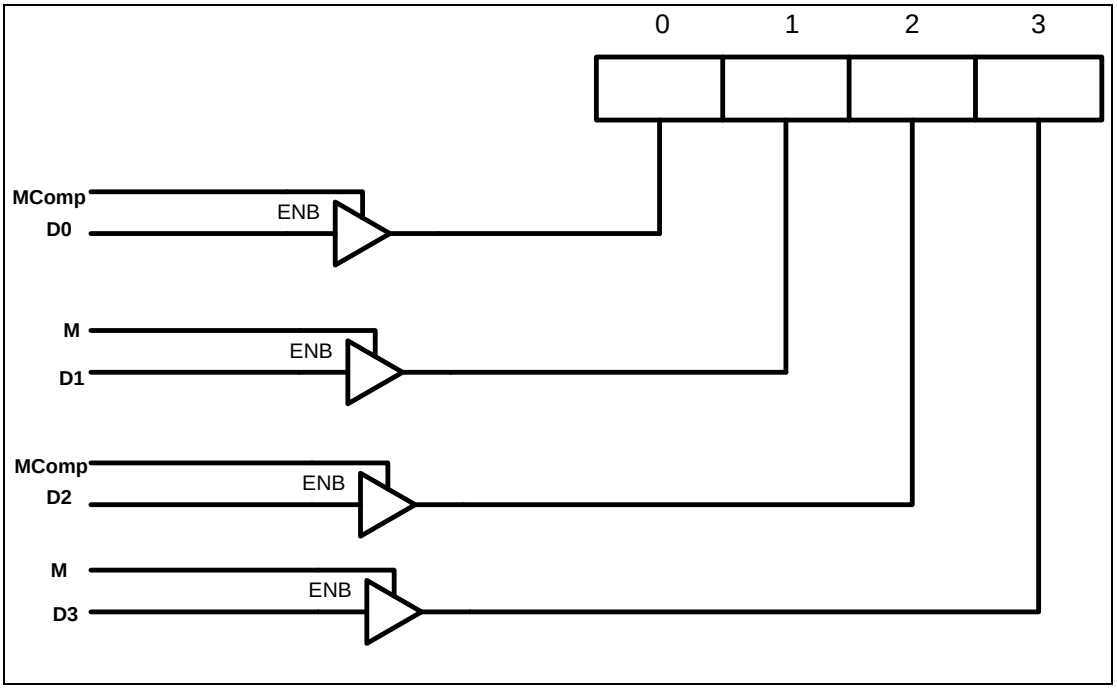

Figure 4.3: Tri-State Buffers for 4-bit Word

This algorithm considers a bit-oriented SRAM so the coupling faults exist between two vertically neighbored cells. Figure 4.4 illustrates the state diagram of two cells in the same column when March C- algorithm is applied. Both cells start with state 00

In modified March C- algorithm, stuck at faults will be detected since each write 0 and write1 operations are followed by reads and they will be applied on each cell. Also, transition faults will be detected as well since each cell will go from 0 to 1 and from 1 to 0. For coupling faults, in order to ensure the same fault coverage of March C- algorithm, each neighboring cells should go through the same transitions shown in figure 4.4. Figure 4.5 illustrates the state diagram of vertically neighbored even cells when modified March C- algorithm is applied.

In the state diagram shown in figure 4.5, it is obvious that some transitions that are available in normal March C- algorithm are missing, such as {00 →01,01→11} and others. Actually this indicates that some cases in coupling faults are missing when modified March C- algorithm is applied. Nevertheless, the new algorithm increases the fault coverage in the rows, so if a word-oriented SRAM is being tested, some coupling faults between horizontally neighbored cells will be detected while normal March C- test detects coupling faults between only two vertically neighbored cells. Hence, totally, the missing fault coverage in coupling faults between cells in the same

column was substituted with coupling faults detected between cells within the same rows. Figure 4.6 shows the state diagram of two horizontally neighbored cells when a modified March C- test is applied. Note that the transitions in two horizontally neighboring cells are just 00→11→00 when normal March C- algorithm is applied.

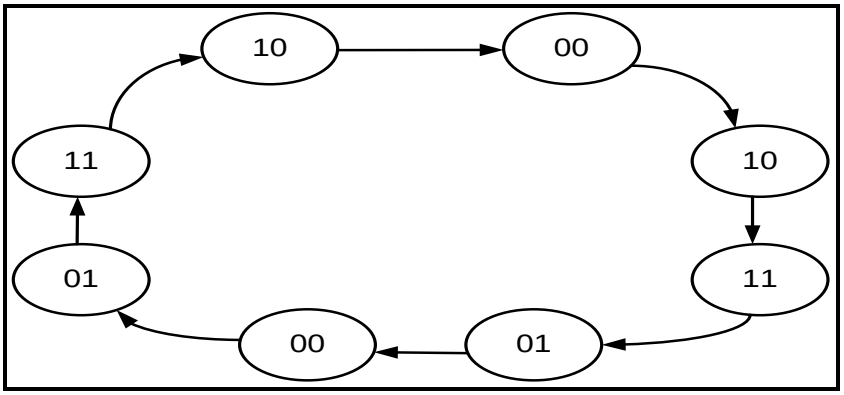

Figure 4.4: Transitions of two vertically neighbored cells in March C-

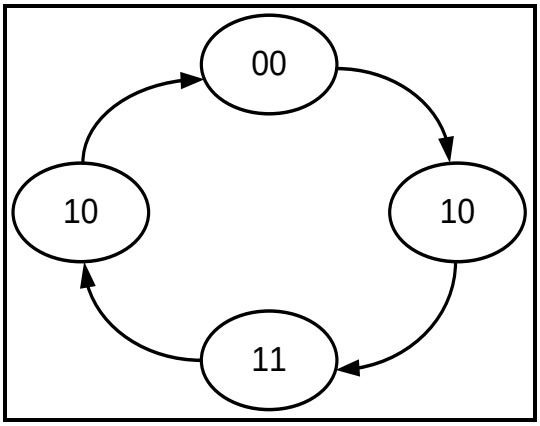

Figure 4.5: States of two vertically Neighbored Cells in Modified March C-

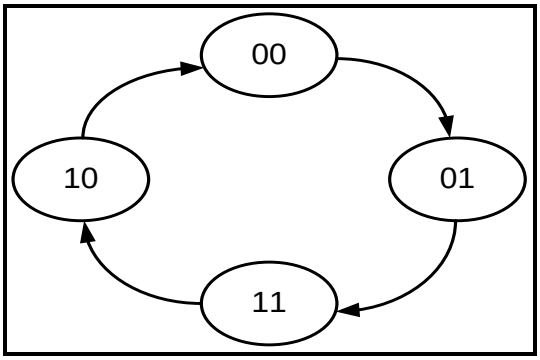

Figure 4.6: States of two horizontally neighbored cells in Modified March C-

To maximize the fault coverage so that coupling faults are detected in all rows and columns, the algorithm has to be expanded as shown in figure 4.7. This **Expanded algorithm** reduces peak power as in modified March C- algorithm, but it

has an average power which exceeds slightly the average power of normal March C- algorithm due to RES operations as shown in (4.6).

↕(W0x,Wx0);↑(R0,Wx1);↑(R01,W1x); ↑ (R1,Wx0); ↑(R10,W0x);↓(R0,W1x); ↓(R10,Wx1); ↓(R1,W0x), ↓(R01,Wx0), ↕(R0)

Figure 4.7: Expanded March C- Algorithm

$$Avg_{expanded} = (9p * P(R) + 10 * (p/2) * [P(W) + P(R)])/19 \quad (4.6)$$

Where $Avg_{expanded}$ represents the average power of expanded March C- algorithm, p is the word length of MUT, P(R) and P(W) represent read and write power respectively . The total number of operations in this test is 19. It is important to note that RES operations form additional power dissipation if compared with normal March C- test but it is not significant since read operation power less than that of write operation and it will be neglected if P(R)<<P(W).

Figures 4.8 and 4.9 show the state diagrams of two vertically and two horizontally neighboring cells respectively when expanded test is applied. It is clear that all transitions included in normal March C- algorithm are found in the expanded test which means that coupling faults will be detected in all rows and columns.

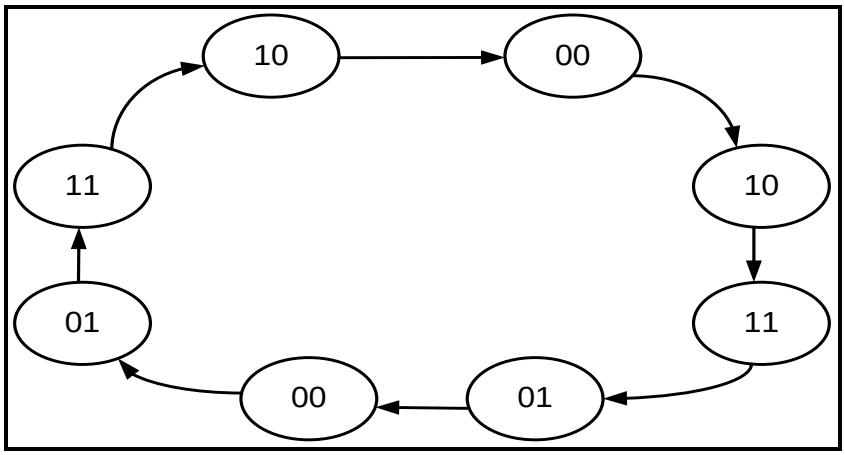

Figure 4.8: States of two vertically neighbored cells in Expanded Test

4.5 Experimental Results

To evaluate the proposed algorithms, peak and average powers were calculated for March C-, modified March C- and the expanded algorithm. Different

MUT configurations were used. C under Linux was used in calculating the power of these algorithms.

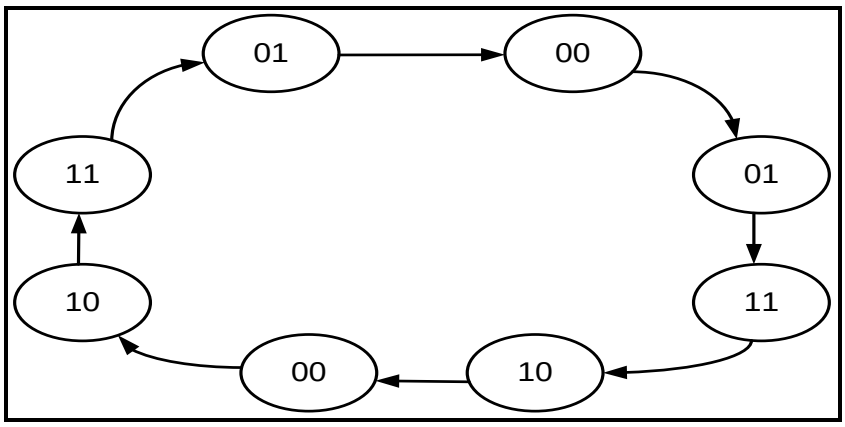

Figure 4.9: Transitions of two horizontally neighbored even cells in Expanded Test

4.5.1Power Estimation

A number of simulations were performed in Cadence Virtuoso Spectre lab to estimate power dissipation during read and write operations. With considering 6T SRAM modeled in 90nm technology with a voltage supply of 1V, read and write powers were estimated to be 72.27µW and 481µW respectively. UMC library was used in the experiments with assuming the widths of transistors (Q3, Q4), (Q1,Q2) and (Q5,Q6) shown in figure 1.3 to be the same. The used transistor models were P_10 SP and N_10 SP from the UMC 90 nm library. Those obtained results of read and write powers were used in power calculations in this dissertation [47].

4.5.2Simulation Results

Table 4.1 shows average and peak power for Normal, modified and expanded March C- tests. Results clearly show that reducing number of cells in the word on which write operation is applied will effectively reduce peak power. Average power is also reduced in case of modified algorithm since the weight of write operation is reduced. In expanded test, average power is closed to that for normal March C- algorithm and it is slightly larger due to read equivalent stress operations. Expanded algorithm peak power is equivalent to that of modified March C- test since write

operation is applied to one cluster only. If the fault coverage has to be maximized, expanded March C- algorithm is recommended to be used with taking into consideration that it entails more testing time.

Table 4.1: Power results of normal, modified and expanded March C- Tests

MUT Configuration	Normal March C-		Modified March C-		Expanded March C-	
	Average Power (mW)	Peak Power (mW)	Average Power (mW)	Peak Power (mW)	Average Power (mW)	Peak Power(mW)
32x32	9.85232	15.3920	5.87974	8.85232	9.9132	8.85232
64x64	18.0420	30.7840	11.7595	17.7046	18.3240	17.7046
64x128	35.2093	61.5680	23.5190	35.4093	35.3128	35.4093
64x256	71.8186	123.1360	47.0736	70.8186	71.9922	70.8186
128x512	142.1253	246.2720	94.0758	141.637	142.3427	141.637

4.6 Summary

In this chapter, a new algorithm was proposed to reduce peak and average power consumed when March tests are applied on MUT. The algorithm is based on dividing the word of MUT into two clusters so that write operation is applied on one cluster only. March C- algorithm was modified based on the new algorithm which is applicable to be used with other tests. To maximize the fault coverage, the proposed scheme was expanded. Results show that peak and average power were reduced effectively when March C- algorithm was modified.

CHAPTER 5

Low Power Schemes for Parallel Testing of Embedded Memories

5.1 Overview

SoC contains a large number of embedded memories with different configurations. Some applications contain memories with wide data words and small address lengths while others contain small data words [48].

Most of low power techniques of SRAM testing focus on reducing the test power of each individual memory and doesn't take into consideration that hundreds of embedded memories will be tested simultaneously resulting in high peak power that may damage the chip if it exceeds the power constraint.

It was proven in section 4.2 that the write current is proportional to the word length of MUT since the data background is written to all bits in the word during write operation. If N memories need to be tested in parallel, it should be taken into consideration that this group contains memories with small word lengths and others with wide word lengths. Thus, dividing those N memories into two clusters, as described in skew scheme in section 2.5, without taking into consideration this factor, may result into large peak power since wide word memories may be in the same cluster and they will be tested in parallel.

In this chapter, two approaches for parallel memory testing at low power consumption will be discussed; **One-stage** approach, wherein only two clusters of memories are considered at a time, and **Multi-stage** approach, wherein multi memory clusters are generated in order to prevent exceeding power constraint. Choosing appropriate clustering when MBIST engine has to be used may impact the placement and the routing of the chip. Hence, an appropriate architecture has to be selected.

Section 5.2 proposes One-Stage scheme whereas Multi-stage scheme is proposed in section 5.3. Appropriate MBIST architecture to avoid routing problem is described in section 5.4. Section 5.5 reports simulation results. Finally, conclusions are proposed in section 5.6.

5.2 One-Stage Scheme

One-Stage scheme, as shown in figure 5.1, is based first on grouping the to-be-tested memories in parallel into two clusters; the assumption is that the total power consumed will not exceed the power constraint; otherwise, Multi-stage scheme (section 5.3) has to be used.

One-Stage Scheme groups the memories into two clusters so that maximal balancing is achieved between those clusters in terms of word size of memories. For example, if there are M memories having word length Wm and N memories having word length Wn, then, these memories are distributed in two clusters so that each cluster will consist of M/2 + N/2 memories.

After the clustering phase, One-Stage scheme tests the two groups in parallel. It is worth nothing that the testing of the second cluster starts one clock cycle later than the testing of the first cluster as shown in figure 5.1. The reason behind this is the appropriate scheduling of test operations. The purpose is to have write operations applied to memory instances of one cluster and read operations simultaneously applied to the second cluster. In this way, the consumed power will be optimal.

```
/*One-Stage Scheme */
begin
1   W={W1,W2,.....,Wn} // Set Of Word Lengths.
2   M=Mw1 U Mw2 U.....U Mwn // Set Of memories where
Mwi  is the set of memories having the word length wi

3   for i=1,2,....,n
4    Add Mwi/2 to Cluster C1
5    Add Mwi/2 to Cluster C2
6   end Loop
7   Start Testing Cluster C1
8   Wait For One clock cycle
9  Start Testing cluster C2
end
```

Figure 5.1: One-Stage Scheme

Let W denotes the set of word lengths in the SoC so that W={W1,W2, …,Wn}, and let Mw1 memories have word length W1, Mw2 have word length W2 and so on. If parallel testing is applied on those memories, the peak power ($Peak_{parallel}$) can be represented in (5.1).

$$\boldsymbol{Peak_{parallel} = \sum_{i=1}^{n} Mwi * Wi * P(W)} \qquad (\mathbf{5.1})$$

Where P(W) represents the power of write operation. This equation clearly shows that when parallel testing is applied to a set of memories, there will be concurrent write operations, and each write operation is applied to the entire word, thus, peak power is the summation of all these simultaneous write operation powers applied on all memories.

When skew scheme is applied, memories are divided into two clusters without considering their word lengths. The main problem related to this scheme is when wide word memories are concentrated in one cluster resulting in large peak power that may damage the chip. Hence, One-Stage scheme ensures that maximal balancing is achieved in the word lengths of memories when they are distributed among clusters. After clustering, write operation will be applied on memories in the first cluster while read operation is applied on memories in the second cluster .Peak power of One-Stage scheme ($Peak_{one\text{-}stage}$) can be represented in equation (5.2).

$$\boldsymbol{Peak_{one-stage} = \sum_{i=1}^{n} (Mwi/2) * Wi * P(W) + \sum_{i=1}^{n} (Mwi/2) * Wi * P(R)} \qquad (\mathbf{5.2})$$

Where P(R) represents the read power operation. With some simplifications, this equation can be re-written as shown in (5.3)

$$\boldsymbol{Peak_{one-stage} = (1/2) * \sum_{i=1}^{n} Mwi * Wi * [P(W) + P(R)]} \qquad (\mathbf{5.3})$$

As shown in (5.3), write operation is applied to half of the cells while read operation is applied to the other half. It is important to remember that always P(R) is less than P(W) and theoretically if P(R) << P(W) then 50% reduction in peak power can be achieved in One-Stage scheme if compared with parallel testing.

5.3 Multi-Stage Scheme

One-Stage scheme reduces peak power effectively with just one additional clock cycle in parallel testing time. Nevertheless, if the chip contains large number of embedded memories with wide word widths, then this scheme may also exceed the power constraint, hence, more than one stage of clustering may be required. In general, power constraint is the chairman in selecting the appropriate scheme.

If multi-stage scheme has to be applied on embedded memories, those memories will be grouped in two clusters as done in one-stage scheme. After that, each cluster will be divided into two other clusters and so on. Figure 5.2 illustrates two-stage clustering process. More clustering may be required according to the power constraint of the chip. If M memories have word length Wm and N memories have word length Wn, then by applying two-stage clustering, four clusters will be obtained {C11, C12, C21, C22} and each of them contains M/4+N/4 memories. After the clustering phase, clusters are grouped into pairs which are called **testing units.** The clusters of any testing unit such as C11, C12 are tested in parallel with one clock cycle skew between them. The testing units are tested sequentially. Consequently, peak power will be reduced to the half while the testing time will be doubled. Hence, some optimization is required.

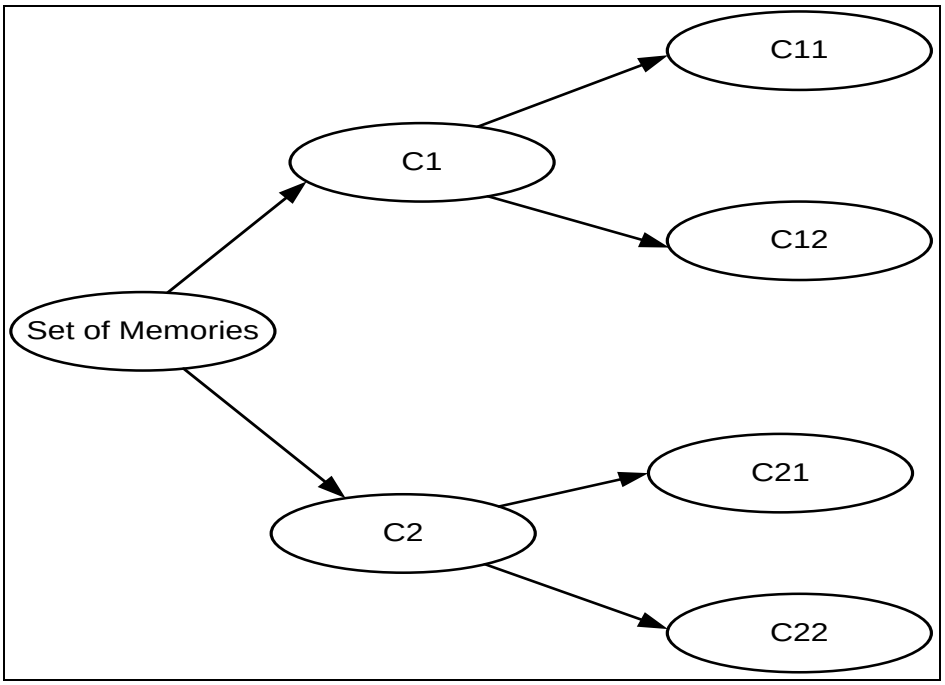

Figure 5.2: Two-Stage Clustering

Figure 5.3 illustrates Multi-Stage Scheme. This scheme consists of three phases:

1. **Clustering Phase:** In this phase, embedded memories in SoC are grouped into clusters so that maximal balancing is achieved in word lengths of memories. First, two clusters are generated and the expected peak power of applying One-Stage scheme is calculated. If it exceeds power constraint, then more clustering is applied. The expected peak power is calculated periodically for each testing unit. When this expected power becomes less than power constraint, clustering phase stops.

2. **Movement Phase:** After clustering phase, memories with maximal address bus length are moved to one testing unit in order to reduce the testing time. For example, if two testing units contain memories with 32, 64 and 128 memory locations, then if they are tested sequentially, the testing time will be 128N+1 for each of them, where N is the number of operations in the applied testing algorithm and one is added for skew. Totally, the testing time will be (128N+1)*2. To reduce this testing time, memories with 128 locations are moved to one testing unit so that the testing time will be (128N+1) + (64N+1). After each movement, expected peak power in each testing unit is calculated and compared with the power constraint.

3. **Testing Phase:** Finally, when expected peak power becomes closer to the power constraint of chip, movement phase stops and testing units are tested sequentially. In each unit, the two clusters are tested with one clock cycle skew between them to satisfy the scheduling described in section 5.2

It is clear that the power constraint of the chip is the key factor in Multi-stage scheme. The expected peak power (which is the peak power of applying One-Stage scheme on a testing unit) has to be calculated and compared periodically with power constraint to decide whether to perform more stages or not. Generally, the peak power of Multi-Stage scheme ($Peak_{multi\text{-}stage}$) can be represented in (5.4).

$$\boldsymbol{Peak_{multi-stage} = (1/K) * \sum_{i=1}^{n} Mwi * Wi * [P(W) + P(R)]} \qquad (5.4)$$

Where K is the total number of clusters generated, Mwi is the number of memories having word length Wi, P(W) and P(R) represent the power of write and read operations respectively. The main problem of this scheme is related to the testing time since the testing units are tested sequentially. Testing time of Multi-Stage scheme (T) can be represented in (5.5) when considering n testing units. It is important to remember that each testing unit consists of two clusters that will be tested in parallel with one clock cycle skew between them.

$$\boldsymbol{T = \sum_{i=1}^{n} (Maxadd_i * N + 1)} \qquad (5.5)$$

Where $Maxadd_i$ represents the number of locations in the memory that has the largest address space in testing unit number i, N is the number of operations in the March test applied and 1 is added for skew. To reduce this testing time, movement phase is applied to add the memories with largest address spaces in one testing unit. Power constraint should be calculated to ensure keeping the peak power below the power constraint of the chip.

```
/*Multi-Stage Scheme */
begin
1   M={M1,M2,.....,Mn}    //Set Of Memories
2   W={W1,W2,.....,Wn}    //Set of Word Lengths
3   A={A1,A2,.....,An}    //Set of Address Lengths
4   Define PowerConstraint,PeakPower;
5   Define upperlimit=Powerconstraint-somevalue;
6    while(peakpower>PowerConstraint)
7      Divide_Into_clusters_With_Balancing(M,W);
8    Calculate_Expected_Peak_Power_For_One-Stage-
Scheme_For_One_Testing_Unit();
9     endLoop
10  while (PeakPower<upperlimit)
11 find_max_address_For_Each_Testing_Unit(A,testingUnit);
12     Move_Memories();
13    Calculate_Expected_Peak_Power_For_One-Stage-
Scheme_For_One_Testing_Unit();
14      endLoop
15    for each Testing Unit
16         Start_Testing_With_Skew();
17         endLoop;
end
```

Figure 5.3: Multi-Stage Scheme

5.4 MBIST Implementation

One of the important issues for the proposed schemes is the architecture that has to be used. If all memory cores in one cluster are connected to one BIST engine, then, this will increase the routing of the system. Thus, BIST engine may become a hot spot. Usually, routing is proportional to the wiring and placement in design [49]. To avoid this serious problem, proposed schemes can be implemented using the architecture shown in figure 5.4. In this architecture, each MUT is connected to a TPG and usually multiple TPGs share one sequencer and they are divided into several groups based on the power constraint. So this feature will be useful since memories will be distributed into groups based on their word lengths as described in the proposed schemes. Usually, in this architecture, memory cores within the same group are tested in parallel so two groups can be tested in parallel with one clock cycle skew between them. This can be done by the appropriate configuration of controller. It was proven that this architecture minimizes routing area overhead by positioning the TPG near each memory, also this is achieved by using serial interface between controller and sequencer, and between sequencer and TPGs [50].

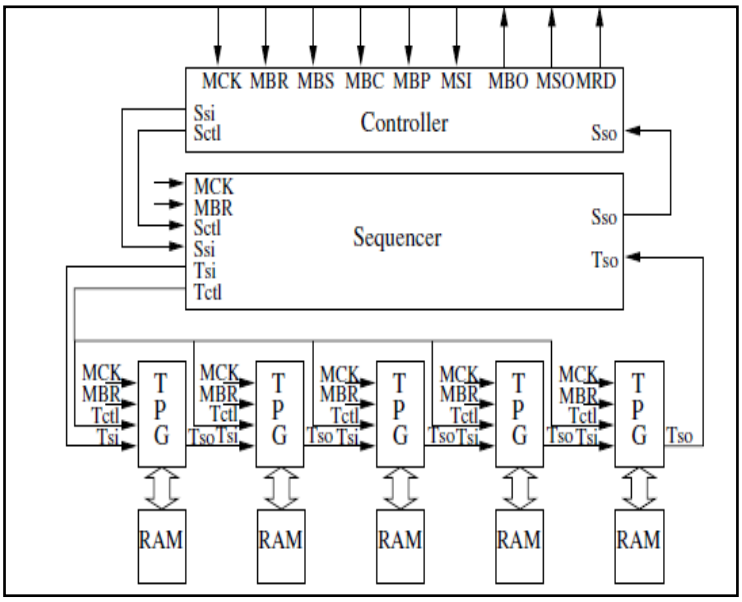

Figure 5.4: Architecture of Low Routing MBIST [50]

In this architecture, the testing pattern (such as element M1 in March C-) is sent from controller to the sequencer. The sequencer generates a command using a register and sends it to other TPGs. Usually the group ID is defined in the command

in order to determine which group will be tested. After that, results are obtained from each MUT and compared with the expected results in the controller.

5.5 Experimental Setup and Simulation Results

Usually embedded memories that are used in different applications have word lengths vary from 32-512 and 640 bits in some cases. For this reason, different memory configurations were used in order to evaluate the proposed schemes. Various numbers of memory cores were used. Table 5.1 shows the memory configurations and their numbers for the tests used in this section. For example, in test 1, the used SoC contains 100 memory cores; 40 of them are 32x32, 20 of them are 64x64, 20 memory cores have the size 64x128, and the other 20 cores have the sizes 128x512 and 128x640.

Table 5.1: Used Memory Configurations

Test	Core.No	Used Memory Configurations					
		32x32	64x64	64x128	64x256	128x512	128x640
1	100	40	20	20	0	10	10
2	200	40	40	40	40	20	20
3	300	60	60	60	60	40	20
4	400	100	100	50	50	50	50
5	500	100	100	100	100	50	50
6	600	200	100	50	50	100	100
7	700	200	100	100	100	100	100
8	800	200	100	100	150	150	100
9	900	150	150	150	150	150	150
10	1000	200	200	200	200	100	100

To apply the proposed schemes, 10 tests were performed. In each test, four schemes were used in order to test the memory configurations shown in table 5.1. These schemes are: Parallel, skew, One-Stage and Multi-Stage schemes. For example, in test 1, 100 memories were tested based on each scheme. Normal March test (in which each read and write operation is applied to the entire word) was considered in testing. Table 5.2 shows the obtained results. Peak power and testing time were calculated for each test and the saving percentage in peak power was calculated if each of the schemes is compared with parallel testing. C under Linux was used in simulating each of the schemes and power values described in section 4.5.1 were used in this section. In table 5.2, N refers to the number of operations in the March test. For example N=10 for March C- algorithm.

Table 5.2: Peak Power and Testing Time for Different Schemes

Test	Parallel		Skew			One-Stage Scheme			Multi-Stage Scheme		
	Power (mW)	Time (Cycles)	Power (mW)	Time (Cycles)	%	Power (mW)	Time (Cycles)	%	Power (mW)	Time (Cycles)	%
1	6156.80	128N	5327.03	128N+1	13	3540.93	128N+1	42	2124.56	192N+`2	65
2	20317.44	128N	17701.57	128N+1	12	11685.06	128N+1	42	6373.67	192N+2	68
3	29860.48	128N	25936.67	128N+1	13	17173.50	128N+1	42	9206.41	192N+2	69
4	41558.40	128N	37634.59	128N+1	9	23901.26	128N+1	42	15934.176	192N+2	61
5	50793.60	128N	44253.92	128N+1	12	29212.66	128N+1	42	15934.18	192N+2	68
6	70803.20	128N	65571.46	128N+1	7	40720.67	128N+1	42	31868.35	192N+2	54
7	80038.40	128N	72190.78	128N+1	9	46032.06	128N+1	42	`31868.352	192N+2	60
8	`98508.80	128N	88045.31	128N+1	11	56654.84	128N+1	42	38950.20	192N+2	60
9	117748.80	128N	104015.47	128N+1	11	67720.25	128N+1	42	47802.53	192N+2	59
10	101587.20	128N	88507.84	128N+1	12	58425.31	128N+1	42	31868	192N+2	68

Results in table 5.2 show that parallel testing for multiple memory instances in SoC results in abrupt increase in the peak power that will exceed the power constraint and may damage the chip. Skew scheme reduces peak power effectively with only one additional clock cycle. The skew scheme was implemented by dividing memory instances in two clusters. The memory instances were added in those clusters randomly without taking into consideration their word lengths. One-Stage Scheme is more efficient in reducing the peak power. The obtained results prove that distributing memories based on their word lengths is a good enhancement of the skew scheme since those memory instances with wide word widths that form the major part of power dissipation during testing are not concentrated in one cluster. The saving percentage in this scheme is 42% if it is compared with parallel testing. Multi-stage scheme takes into consideration both the peak power and the testing time. But power constraint is the key factor in this algorithm. Results show that using multi-stage scheme will reduce the peak power effectively but this will increase the testing time as well. In general, using Multi-stage scheme will be useful in applications wherein the testing time is not critical.

To achieve more peak power saving, Modified March C- algorithm was combined with One-Stage scheme and applied on the same tests shown in Table 5.1. Results in table 5.3 show that combining this algorithm with One-Stage scheme will result in power saving up to 60% if compared with the parallel testing with negligible cost in the testing time.

Table 5.3: Combining One-Stage Scheme with Modified March C- Algorithm

Test	One-Stage Scheme with Normal Mach C-		One-Stage Scheme with Modified March C-	
	Peak(mW)	Saving (%)	Peak(mW)	Saving(%)
1	3540.93	42	2232.9920	63
2	11685.06	42	7368.8736	63
3	17173.50	42	10830.0112	63
4	23901.26	42	15072.6960	63
5	29212.66	42	18422.1840	63
6	40720.67	42	25679.4080	63
7	46032.06	42	29028.8960	63
8	56654.84	42	35727.8720	63
9	67720.25	42	42705.9720	63
10	58425.31	42	36844.3680	63

5.6 Summary

In this chapter, a new scheme was proposed to reduce peak power when parallel testing is applied to multiple embedded memories in SoC. This scheme is based on grouping memories in clusters based on their word lengths and scheduling read and write operations. Thereafter, the scheme was generalized in Multi-Stage scheme to be applicable for applications with low power constraint. Finally, it was proven that very good reduction in peak power was achieved when the proposed scheme is combined with modified March C- algorithm. To avoid routing overhead, a good architecture was selected from literature to be suitable for MBIST implementation.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

Testing power is a principal contributor for power dissipation of embedded SRAMs in SoC. High switching activities and simultaneous write operations during parallel testing are the main sources of this power dissipation. It was proven that power consumed during testing could be twice the power consumed during the functional mode. Although many techniques were developed for memory testing, only few of these techniques dedicated for reducing testing power.

This dissertation proposed a number of techniques to reduce testing power of embedded memories. These techniques target different categories of applications.

For on-line testing and personal devices which are tested for stuck-at faults, Zero-One algorithm was enhanced so that the switching activity during testing is minimized. This is done by reducing switching activity while address decoders are accessed. Then, the test was reordered so that the switching activity while accessing write drivers is reduced. Results show that applying the pattern {↕(W0,R0); ↕(W1,R1)} will cause the least switching activity in data bus and using DS-LFSR with BS-LFSRs for its slow and normal parts has the least switching activity in address decoder. These results were introduced in chapter 3.

When memory has to be intensively tested, it goes through different tests. March algorithms are commonly used in such a test. It is infeasible to use LFSR as an address generators for March tests since the order of addresses generated should be taken into consideration. Chapter 4 proposes an enhancement on March tests to reduce peak and average power. This is done by dividing the word of MUT into two clusters so that write is applied to one cluster at any moment of time. The idea behind

this approach is the fact that write power is proportional to the word length of MU”T. By this way, modified March C- algorithm was proposed. It is important to note that dividing word into clusters is applicable for other tests. Finally, to maximize the fault coverage, Modified March C- algorithm was expanded. The main disadvantage of such expansion is the large testing time.

Usually when parallel testing is applied to a number of memories in SoC, simultaneous write operations result in high peak power that may damage the chip if exceeds power constraint. Hence, a good management of parallel testing was proposed in chapter 5. The proposed One-Stage scheme is based on grouping memories in two clusters so that maximal balancing is achieved in their word lengths, then, read and write operations are scheduled so that concurrent write operations between clusters are avoided. This scheme was then generalized into Multi-Stage scheme to cover applications with low power constraint. In general, power constraint is the chairman in selecting the appropriate scheme. When One-Stage scheme was combined with Modified March C- algorithm, up to 60% reduction in peak power was achieved with negligible cost in the testing time.

6.2 Future Work

One of the possible future works on this dissertation is to find a low power and low hardware area address generator which is suitable to be used with March tests. Using this generator with low power March tests, such as those generated by genetic algorithm or PSO scheme, will reduce dynamic power dissipation effectively.

Another future work is to modify SRAM cell so that read equivalent stress operations are not applied. In such a case, modified March tests will result in 50% reduction in peak power.

References

[1] Semiconductor Industry Association, “International technology roadmap for semiconductors (ITRS), 2005 edition”, Dec. 2005.

[2] N. Noor, Y. Yusof, and A. Sparon, “Low Area FSM-Based Memory BIST for Synchronous SRAM, proceeding of the international colloquium of Signal Processing and Its application, 2009, pp. 409-412.

[3] A. Kokrady, C. Ravikumar, and N. Chandrachoodan, “Layout-Aware and Programmable Memory BIST Synthesis for Nanoscale System-on-Chip Designs”, proceeding of the Asian Test Symposium (ATS), 2008, pp.351-356.

[4] C. Hsu, and T. Chen, “Built-in Self Test Design for Fault Detection and Fault Diagnosis in SRAM-Based FPGA”, Instrumentation and Measurement, IEEE Transactions, 2009, pp. 2300 - 2315

[5] A .Abu-Issa and S. Quigley, “LT-PRPG: Power Minimization Technique for Test-Per-Scan BIST”, proceeding International Conference on Design & Technology of Integrated Systems in Nanoscale Era, 2008, pp.1-5

[6] R. Rajsuman, "Design and Test of Large Embedded Memories: An Overview", IEEE Design and Test of Computers, May 2001, pp. 16-27.

[7] P. Athe, and S.Dasgupta, “A comparative Study of 6T, 8T and 9T decanano SRAM cell”, proceeding the IEEE symposium on Industrial Electronics and Applications, 2009, pp.889-894.

[8] F. Duan,, R. Castagnetti, R. Venkatraman O. Kobozeva , and S. Ramesh, “Design and Use of memory-specific Test Structures to ensure SRAM Yield and Manufacturability, proceedings fourth international symposium on Quality Electronics Design, 2003, pp. 119-124.

[9] R. Joshi, R. Williams; E. Nowak , K. Kim J. Beintner, T. Ludwig,; I. Aller, and C. Chuang, “FinFET SRAM for high-performance low power applications”, proceedings of the 34th European conference on Solid-State Device Research, 2004, pp.69-72

[10] X. Du and X. Zhang, “The impact of memory hierarchies on cluster computing”, proceeding the 13th international symposium on parallel and distributed processing, 1999, pp.61-69

[11] A. Palove, and M. Sachdev, “CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies”, Springer Science and Business Media B.V, 2008 ,pp.15-18

[12] B. Mohammad, S. Bijansky, A. Aziz and J. Abraham, “Adaptive SRAM memory for Low Power and High Yield, IEEE International Conference on Computer Design, 2008, pp.176-181.

[13] R. Keerthi.; and H.Chen, “Stability and Static Noise Margin Analysis of Low-Power SRAM”, proceedings IEEE Instrumentation and Measurement Technology Conference, 2008, pp.1681-1684

[14] V. Goor and I.Tilili, “ March Tests for word oriented memories, proceedings IEEE Design, Automation and Test in Europe, 1998, pp.501-508

[15] R. Gibbins, et al. "Design and Test of a 9-Port SRAM for a 10Gb/s STS1 Switch", IEEE International Workshop on Memory Tech., Design and Testing, pp. 83-87, 2002.

[16] M. Abramovici, M. Breuer and A.Friedman, “ Digital Systems Testing and Testable Design”, Jaico Publishing House, 2006, pp.93-95

[17] N. Haron, S. Junos, and A. Abdul Aziz, “Modeling and Simulation of Microcode Memory Built-in Self Test Architecture for Embedded Memories, proceedings of International Symposium on Communication and Information Technology, 2007, pp. 136- 139.

[18] R. Adams, “High Performance Memory Testing: Design Principles, Fault Modeling and Self-Test, Kluwer Academic Publisher, 2003, pp. 104-139.

[19] X. Fan, W. Moore, C. Hoora and G. Gronthoud, “Stuck-open fault diagnosis with Stuck-at Model”, proceedings European Test Symposium, 2005, pp. 182-187

[20] M. Niamat, M. Lalla, and J. Kim,” Testing Faults in SRAM Memory of Vertex-4 FPGA”, proceeding the IEEE International Midwest Symposium, 2009, pp.965-970.

[21] V. Vardanian and Y.Zorian, “ A March-based fault location for static Random Access Memories”, proceedings IEEE international Online Testing Workshop, 2002,pp.256-261

[22] S. Hamdoui, A. Goor and M. Rodgers, “March SS: A Test for all static simple RAM Faults”, proceedings of the 2002 IEEE international Workshop on Memory Technology, Design and Testing, 2002, pp.95 -100.

[23] D. Papakostas and A. Hatzopoulos, “Detection of time-delay Related faults using Fourier Phase Components of Power Supply Current”, Electronic Letters, 2004, pp. 7-8

[24] N. Haron, S. Junos, A. Abdul Razak, and M. Idris, " Low Area FSM-Based Memory BIST for Synchronous SRAM", proceedings the 5th international Colloquium on Signal Processing & Its Applications, 2009, pp.409-412.

[25] H. Hashempour;, F. Meyer and F. Lombardi," Analysis and Measurement of fault coverage in a combined ATE and BIST environment", proceedings IEEE International Conference on Instrumentation and Measurement, 2004, pp.300-307

[26] Z.Zhang, Z, Wen and L. Chen, "BIST Approach for Testing Embedded Memory Blocks in System-on-Chips", IEEE International Conference on Testing and Diagnosis, 2009, pp.1-3.

[27] W. Wang and K. Lee, "A Complete Memory Address Generator for scan Based March Algorithms", proceedings of the IEEE International Workshop on Memory Technology, Design and Test, 2005, pp.83-88

[28] A. Abu-Issa, "Low Power High Fault Coverage Test Techniques for Digital VLSI Circuits", University of Birmingham, 2009, pp.3-5

[29] N. Haron, S. Junos, A. and Abdul Razak, "Modeling and Simulation of Finite State Machine Memory Built-in Self Test Architecture for Embedded Memories", proceedings the Asia-Pacific Conference On Applied Electromagnetics", 2007, pp. 1-5

[30] L. Wang, C. Stroud, and N. Toubam "System On Chip Test Acrhitectures",Morgan Kaufmann Publishers, 2008, pp.308-339.

[31] W. Collier, "Testing Memory Models", proceedings the 9th International Workshop on Microprocessor Test and Verification, 2008, pp.14-17

[32] Said Hamdoui, and Zaid Al-Ars, "Scan More with Memory Scan Test", proceedings of 4th International DTIS conference, 2009, pp.204-209.

[33] S. Al-Harbi, F. Noor and F. Al-Turjman, "March DSS: A New Diagnosis March Test for All Memory Simple Static Faults", proceedings IEEE Transaction on Computer Aided- Design of Integrated Circuits and Systems, 2007, pp.1713-1720

[34] R. David, "Random Testing of Digital Circuits", Marcel Dekker Inc, 1998,pp.213- 220

[35] S. Singh , S. Azmi, N. Agrawal,.P. Phani, and A. Rout, "Architecture and Design of a high Performance SRAM for SoC design", proceedings of Design and Automation Conference, 2002, pp.447-451

[36] L. Dilillo, P. Rosinger, and B. Al-Hashimi, "Minimizing Test Power in SRAM through Reduction of Pre-charge Activity", proceeding the Design, Automation, and Test in Europe (DATE) conference,2006,pp.1-6

[37] C. Gayathri , N. Kayalvizhi , and M. Malligadevi" Generation Of New Match Tests With Low Test Power And High Fault Coverage By Test Sequence Reordering Using Genetic Algorithm", International conference on Advances in Recent Technologies in Communication and Computing (ARTCom),2009, pp.699-703.

[38] K. Kumar, S. Kaundinya, and S. Chattopadlhay, "Particle Swarm Optimization Based Scheme for Low Power March Sequence Generation for Memory Testing", Asian Test Symposium (ATS), 2010, pp.401-406

[39] Y. Wu, and A. Ivanov, "Low Power SoC Memory BIST", proceeding of the 21st IEEE international Symposium on Defect and Fault Tolerance in VLSI Systems, 2006, pp.197-205.

[40]B. Fang and N. Nicolici, "Power-Constraint Embedded Memory BIST Architecture", proceeding of the 18th IEEE international Symposium on Defect and Faul Tolerance in VLSI Systems (DFT'03), 2003, pp.451-458.

[41] M. Fischerova, T. Pikula, M. Simlastik, A. Bosio, D. Stefano, and G. d.Natale, "A tool for teaching memory testing based on BIST", proceedings the international Baltic Electronic Conference, 2006, pp. 1-4

[42] A. Abu-Issa, and S. Quigley, "Bit-swapping LFSR for low-power BIST", Electronic Letters, 13 March, 2008, pp. 401-402

[43] S. Wang, and S. Gupta, "DS-LFSR: A BIST TPG for Low Switching Activity", IEEE Transactions On Computer-Aided Design of Integrated Circuits And Systems, 7,July, 2002, pp. 842-851

[44] M. Tehranipoor, M. Nourani, and N. Ahmed, "Low Transition LFSR for BIST-Based Applications", Proceedings of the 14th Asian Test Symposium, 2005, pp. 138-143

[45] http://www.xilinx.com/support/download, Xilinx Inc, Xilinx Design Suite Version 12.1

[46] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale and P. Prinetto,"March Test Generation Revealed", proceedings IEEE Transactions on Computers, 2008, pp. 1704-1713.

[47] Virtuoso® Analog Design Environment User Guide, Cadence Design System, Product Version 5.1.41.

[48] A. de Goor, G. Gaydadjiev, and S. Hamdioui, “Memory Testing with a RISC Microcontroller, proceeding the Design, Automation, and Test in Europe (DATE) conference, 2010.

[49] T. Chien, W. Chao, C. Li, Y. Chang, K. Laio, M. Chang, M. Tsai and C. Tseng, “BIST Design Optimization for Large-Scale Embedded Memory Cores”, ICCAD, 2009, pp.197-200

[50] L. Denq, and C. Wu, “A Hybrid BIST Scheme for Multiple Heterogeneous Embedded Memories”, Asian Test Symposium (ATS), 2007, pp.349-354

Appendix A
March Tests

March test is a sequence of read and write operations that are applied to Memory Under Testing (MUT) to detect different types of faults. Usually each March test consists of a number of March elements. Each element is applied to all memory locations. Usually a March test is delimited by { }, whereas each element is delimited by ( ). Different elements are separated by semicolons. The addressing orders of a March test could be increasing (↑), decreasing (↓) or don't care (↕).

To represent the faults detected by a March test, a fault primitive is used. Usually the fault primitive has the form <S/F/R> where S represents the operation that sensitizes the fault, F represents the faulty value of the cell whereas R represents the logical value if the applied operation is read. For example, if a cell has a transition fault (when going from 0 to 1) then the fault primitive will be <0w1, 0,- >. In case of faults involving two faults such as coupling faults, the fault primitive is represented by <Sa;Sv/F/R> where Sa represents the operation or state of aggressor cell and other parts for victim cell. Table A.1 shows a number of faults and their fault primitives. SF represents state faults, TF means transition fault, WDF is write disturb fault. RDF is read destructive fault, DRDF is deceptive read destructive fault and IRF is incorrect read fault.

Table A.1: Fault Primitives

Fault	Fault Primitive
SF	**<1/0/->, <0/1/->**
TF	**<0w1/0/->, <1w0/1/->**
WDF	**<0w0/↑/->,<1w1/↓/->**
RDF	**<r0/↑/1>,<r1/ ↓/0>**
DRDF	**<r0/↑/0>,<r1/↓/1>**
IRF	**<r0/0/1>,<r1/1/0>**

Several March tests where developed in order to detect more faults in the MUT. Table A.1 shows a number of March tests with their operations and fault coverage where SAF represents stuck-at faults, AF means address decoder faults, TF means transition Faults and CF is coupling faults. March C- algorithm was found

mainly to detect coupling faults which occurs due to bridging between two neighboring cells.

Table A.2: Some March Tests with their Fault Coverage

Test	Sequence	N	Fault types detected			
			SAF	AF	TF	CF
MATS	⇕(w0)⇕(r0,w1)⇕(r1)	4N	+	+/-	-	-
MATS+	⇕(w0)⇑(r0,w1)⇓(r1,w0)	5N	+	+	-	-
MATS++	⇕(w0)⇑(r0,w1)⇓(r1,w0,r0)	6N	+	+	+	-
March Y	⇕(w0)⇑(r0,w1,r1) ⇓(r1,w0,r0)⇕(r0)	8N	+	+	+	+/-
March C-	⇕(w0)⇑(r0,w1)⇑(r1,w0) ⇓(r0,w1) ⇓(r1,w0)⇕(r0)	10N	+	+	+	+

Appendix B
Genetic Algorithm

Genetic algorithm is heuristic based algorithm which is used to find an optimal solution for a problem. It belongs to Evolutionary Algorithm (EA) that was found mainly for optimization.

Genetic algorithm starts with an initial population which is defined based on the problem that has to be solved. Then, a number of operations are applied on this population to generate more populations. A fitness function is defined and calculated for each population. Usually, the population with higher fitness function is considered better. The main operations applied on any population in genetic algorithm are:

1. Mutation which is used to generate new populations based on incremental changes. So an incremental change is applied on a population and the new population is accepted if it has a better fitness function

2. Crossover operation which combines two available populations in order to achieve a better solution. Actually, this is the basic operation in genetic algorithm.

Genetic algorithm can be used in order to re-order March test sequences so that the test power is minimized, this can be achieved as following:

1. Given a March test T, the initial population is found, which is the set of all write operations in T.
2. Mutation and crossover operations are applied on the initial population to generate new populations.
3. The switching activity of each newly generated population is calculated and also the fault coverage which is determined by the types of fault that have to be detected. The fitness function is calculated for each newly generated population, the one with the maximum fitness function is selected.

Figure B.1 illustrates the crossover operation applied within genetic algorithm on the populations of a certain March test.

```
Population1:  w0 w1 w1 w0 w0 w1 w1 w0 w0
Population2: w0 w1 w1 w0 w1 w1 w0 w1 w0
Crossover:
              w0 w1 w1 w0 w0  |   w1 w1 w0 w0
              w0 w1 w1 w0 w1  |  w1 w0 w1 w0
after crossover:
       solution 1: w0 w1 w1 w0 w0 w1 w0 w1 w0
       solution 2: w0 w1 w1 w0 w1 w1 w1 w0 w0
```

Figure B.1: Crossover Operation

Appendix C
Particle Swarm Optimization Scheme

Particle Swarm Optimization (PSO) is a computational method that is used to optimize a problem using iteratively trying to improve a candidate solution. This scheme is stochastic since it is based on randomly generated variables in generating new populations.

PSO is based on generating a number of populations; each of them is called a particle. A fitness function is calculated for each particle, then flip operation is applied on each particle to generate new particles which are accepted if they have a better fitness function. Finally, the particle with the best global fitness function is selected as optima.

PSO scheme can be used in finding optimal March tests in terms of power and fault coverage. To achieve this, any particle Pi consists of two sequences: Oi which represents the operations (read and write) and Di which represents the directions. The main operation used in PSO in order to generate other particles is flip operation which decides whether a value in Oi or Di has to be flipped or not based on flipping operator. Usually the flip operator requires flip sequences (FS_{L1} and FS_{L2}) in order to align the local corresponding values of local fitness function. By these sequences, a bit is defined in order to determine whether flipping will happen or not. In general, generation of new March tests based on PSO scheme can be done as following:

1. This scheme takes the number of operations (NoP) in March test as input.
2. A number of particles are generated randomly, each of them consists of NoP operations which includes read and write operations. Usually random generation of these particles is based on some random variables.
3. The fitness function is calculated for each particle.
4. Flip operation is performed and if the newly generated particle has a better fitness function, then it is accepted. This operation is repeated for each particle.
5. The global fitness function is obtained for all particles in order to select the best March test.

Appendix D
Maximal Length LFSR

A maximal length Linear Feedback Shift Register (LFSR) generates all possible testing vectors (except the zero's vector) with 2^n-1 clock cycles. This is achieved by selecting the appropriate location of XOR gate. Usually the characteristic polynomial is used to represent an LFSR and its XOR gate. Figure B.1 shows LFSR with the characteristic polynomial shown in equation (B.1):

$$P(x) = x^8 + x^6 + x^5 + x + 1 \qquad \text{(B.1)}$$

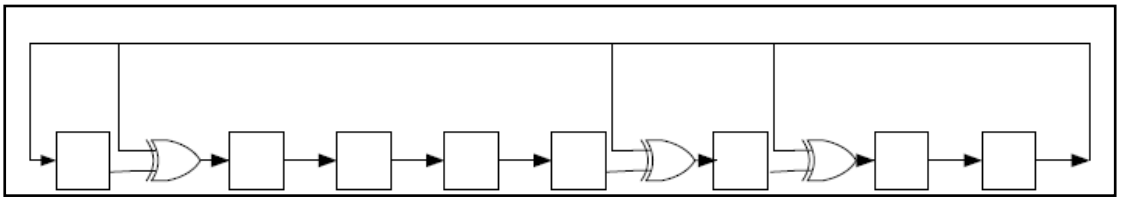

Figure D.1: Maximal Length LFSR

Table D.1 shows the maximal length LFSRs for different sizes. For example, if 9-bit LFSR has to be used, then XOR gates should be located in the inputs of flip flop #9 and flip flop #4.

Table D.1: Maximal Length LFSRs

Degree (n)	Polynomial
2,3,4,6,7,15,22	x^n+x+1
5,11,21,29	x^n+x^2+1
8,19	$x^n+x^6+x^5+x+1$
9	x^n+x^4+1
10,17,20,25,28	x^n+x^3+1
12	$x^n+x^7+x^4+x^3+x+1$
13,24	$x^n+x^4+x^3+x+1$
14	$x^n+x^{12}+x^{11}+x+1$
16	$x^n+x^5+x^3+x^2+1$
18	x^n+x^7+1
23	x^n+x^5+1
26,27	$x^n+x^8+x^7+x+1$
30	$x^n+x^{16}+x^{15}+x+1$